MASTER EN SCIENCES INFORMATIQUES

Développement d'un agent intelligent adapté à un jeu à information imparfaite

Verbeiren, Nicolas

Award date:
2013

Awarding institution:
Universite de Namur

FACULTÉS UNIVERSITAIRES NOTRE-DAME DE LA PAIX, NAMUR

Faculté d'Informatique

Année académique 2012-2013

DÉVELOPPEMENT D'UN AGENT INTELLIGENT ADAPTÉ À UN JEU À INFORMATION IMPARFAITE

Nicolas Verbeiren

Promoteur : _____ (Signature pour approbation du dépôt - REE art. 40)

P^r D^r I^r Pierre-Yves Schobbens

Mémoire présenté en vue de l'obtention du grade de
Master en Sciences Informatiques.

RÉSUMÉ

On savait les jeux être un excellent sujet d'étude pour l'intelligence artificielle. Le poker en est certainement un très bon exemple. Information imparfaite et évènements aléatoires font de ce jeu un problème d'une grande complexité et par conséquent un challenge excitant pour certains scientifiques. Preuve de cet engouement, l'« Annual Computer Poker Competition » organisée depuis 2006 par l'AAAI rassemble des équipes du monde entier et les meilleurs agents de la spécialité.

Ce travail va d'abord s'attacher à décrire la problématique et l'état de l'art du domaine avant de se concentrer sur l'implémentation d'un agent en vue de participer à l'ACPC 2013.

Mots clés

Poker ; Jeux à information imparfaite ; Équilibre de Nash; Théorie de jeux; Apprentissage supervisé

Abstract

Games are known to be an excellent subject for artificial intelligence. Poker is certainly a very good example. Imperfect information and random events make this game a complex problem and therefore an exciting challenge for some scientists. Demonstration of this interest, the "Annual Computer Poker Competition" organized since 2006 by the AAAI brings together teams from around the world and the best agents of the specialty.

This work will first attempt to describe the problem and the latest developments in the field before focusing on the implementation of an agent to participate in the 2013 ACPC.

Keywords

Computer poker; Imperfect information games; Nash equilibrium; Game theory; Supervised learning

AVANT-PROPOS

Ingénieur en électromécanique de formation, je suis fasciné, depuis le milieu des années 90 et les fameuses confrontations entre "Deep Blue" et Garry Kasparov, par l'intelligence artificielle.

Malheureusement mon parcours professionnel ne m'a jamais permis d'approcher ce domaine et j'ai récemment décidé, avec l'intention de remédier à ce manque, de commencer des études d'informatique.

C'est donc naturellement vers l'intelligence artificielle que je me suis tourné pour le choix de mon sujet de mémoire.

J'ai été surpris d'apprendre qu'un jeu comme les « Dames » avait déjà été résolu depuis 2007 par un chercheur de l'Université d'Alberta au Canada.

Cette même Université a depuis lors concentré ses recherches en IA sur le poker et participe chaque année à l'ACPC, une compétition internationale de « Computer Poker ». Ce sujet m'a immédiatement passionné et malgré l'image quelque fois peu glorieuse de ce jeu, sa complexité en fait un excellent thème d'étude.

Outre la complexité, l'autre intérêt de l'application de l'IA au poker est l'incroyable diversité des matières concernées (informatique, mathématiques, sciences économiques, sciences cognitives).

Je tiens à remercier mon promoteur, le professeur Pierre-Yves Schobbens, bien sûr pour son précieux conseil et son suivi tout au long de l'élaboration de ce mémoire mais également pour sa grande ouverture d'esprit qui m'a permis d'étudier ce sujet quelque peu atypique.

Je tiens également à remercier les membres du jury pour avoir accepté d'examiner ce travail ainsi que mon collègue Serkan Eryilmaz pour son aide et notamment ses précieux éclaircissements sur de nombreux sujets.

Enfin, je tiens tout particulièrement à remercier mes parents pour … tout !

TABLE DES MATIÈRES

TABLE DES ILLUSTRATIONS

Figures

Tables

Équations

INTRODUCTION

Depuis les prémices de la théorie des jeux, les théoriciens de la discipline que sont Émile Borel[2], John von Neumann[13] et John Nash[19] ont tous montré un intérêt notable pour le poker. Bien que le poker dans sa version Texas Hold'em soit devenu extrêmement populaire depuis les années 2000, l'intérêt actuel des scientifiques pour ce jeu tient néanmoins plus à ces caractéristiques intrinsèques que sont la présence d'informations cachées et de composantes aléatoires qui en décuple la complexité.

Un cadre précis définit par des règles simples, permettant une modélisation fidèle, font du jeu un terrain d'expérimentation idéal pour l'intelligence artificielle.

La victoire, en 1997, de « Deep Blue » contre Garry Kasparov (première victoire d'un ordinateur contre un champion du monde d'échecs en titre) a mis un coup de projecteur sur le domaine de l'IA appliqué au jeu mais il ne s'agit-là que d'un haut fait parmi d'autres :

- Au Backgammon, l'ordinateur prendra le dessus sur les meilleurs joueurs du monde dès les années 1980.
- Aux Dames, Chinook[21] le programme développé par Jonathan Schaeffer remporte le championnat du monde en 1994.
- A Othello, le programme Logistello[3] a battu en 1997 le champion du monde humain par 6 jeux à 0.
- Au scrabble, jeu bien plus complexe qu'il n'y parait, le programme Quackle a battu en 2006 un ancien champion du monde humain.

Les recherches continues pour des jeux comme le Go ou le poker dont la complexité représente un challenge encore bien supérieur.

Afin de dynamiser les recherches sur le poker, l'association Américaine d'Intelligence Artificielle (AAAI) organise depuis 2006 une compétition de

Poker (Annual Computer Poker Competition - ACPC), destiné au robot, se déroulant dans le cadre de leur conférence annuelle.

L'objectif de ce mémoire est d'acquérir le savoir nécessaire dans le domaine afin de pouvoir participer à cette compétition.

Afin, très modestement, d'apporter ma pierre à l'édifice nous avons convenu mon promoteur et moi d'utiliser une technique qui, à notre connaissance, n'avait jusqu'ici pas été testée sur le poker, en l'occurrence les « modèles de Markov de maximum entropie » (MEMM).

Ce mémoire se divise en deux, une première partie « théorie » s'attache à analyser le poker à travers différentes disciplines. Cette partie présente également l'état de l'art du domaine ainsi que le cadre théorique utilisé ultérieurement. La seconde partie « pratique » décrit le développement d'un agent avec pour objectif de participer à l'ACPC 2013.

PARTIE 1 : THÉORIE

1 LE JEU

1.1 Modèle

Le poker est un ensemble de jeux de cartes de formes variées. On peut néanmoins dégager une série de caractéristiques communes décrites ci-dessous.

1.1.1 Assistance

Le nombre minimum et maximum de joueurs autorisés dans une partie ; celui-ci est compris entre 2 et 10 joueurs. Des termes spécifiques existent pour certaines configurations telles que :

- Full ring (9 ou 10 joueurs),
- Heads-up (2 joueurs),
- Short Handed (± 6 joueurs).

1.1.2 Univers

Nous appellerons univers l'ensemble des cartes à jouer utilisées dans le jeu.

Les jeux de poker utilisent une partie, la totalité ou même parfois plusieurs jeux de 54 cartes. L'enseigne, désigne la marque de reconnaissance des quatre séries le Carreau ♦, le Cœur ♥, le Trèfle ♣ et le Pique ♠. Par souci de clarté nous adopterons 4 couleurs à la place des deux utilisées généralement.

Nous utiliserons la notation ci-dessous pour nommer les éléments du jeu de carte.

	2	3	4	5	6	7	8	9	T	J	Q	K	A	W
Heart ♥	*2h*	*3h*	*4h*	*5h*	*6h*	*7h*	*8h*	*9h*	*Th*	*Jh*	*Qh*	*Kh*	*Ah*	*Wh*
Spade ♠	*2s*	*3s*	*4s*	*5s*	*6s*	*7s*	*8s*	*9s*	*Ts*	*Js*	*Qs*	*Ks*	*As*	*Ws*
Diamond ♦	*2d*	*3d*	*4d*	*5d*	*6d*	*7d*	*8d*	*9d*	*Td*	*Jd*	*Qd*	*Kd*	*Ad*	
Club ♣	*2c*	*3c*	*4c*	*5c*	*6c*	*7c*	*8c*	*9c*	*Tc*	*Jc*	*Qc*	*Kc*	*Ac*	

Tableau 1 : Convention de nommage des cartes du jeu de 54 cartes

1.1.3 Système

Le système, permettant de départager les joueurs, est commun à tous les jeux de poker et représente l'ensemble des combinaisons de 5 cartes et leur valeur respective. Il est basé sur deux paramètres :

1. ***Le rang***

Dans l'ordre croissant : 2, 3, 4, 5, 6, 7, 8, 9, T, J, Q, K, A

L'As (*A*) est un cas particulier, il peut parfois être indifféremment considéré comme la carte la plus forte ou la plus faible.

2. ***L'enseigne***

Les enseignes ne possèdent pas de valeur respective : ♦ ≈ ♥ ≈ ♣ ≈ ♠

Le système peut être généralement synthétisé en 8 catégories listées dans le tableau ci-dessous :

Nom "fr"	Nom "en"	Min.	Max.
hauteur	high-card	*7d 5d 4s 3c 2c*	*Ah Kc Qd Js 9s*
paire	pair	*5d 4d 3s 2s 2c*	*Ah Ac Kd Qs Js*
double paire	two pair	*4h 3d 3s 2s 2c*	*Ah Ac Kd Ks Qs*
brelan	three of a kind	*2s 2d 2c 3h 4d*	*As Ad Ac Kh Qd*
suite	straight	*Ac 2s 3s 4h 5h*	*Tc Js Qs Kh Ah*
couleur	flush	*7d 5d 4d 3d 2d*	*Ad Kd Qd Jd 9d*
main pleine	full house	*2s 2d 2c 3h 3d*	*As Ad Ac Kh Kd*
carré	four of a kind	*2s 2d 2c 2h 3d*	*As Ad Ac Ah Kd*
Quinte flush	straight flush	*Ac 2c 3c 4c 5c*	*Tc Jc Qc Kc Ac*

Tableau 2 : Catégories de combinaisons

Dans de rares jeux de poker, d'autres catégories sont utilisées ; ceci est notamment dû à l'utilisation de plusieurs jeux de carte ou de jokers (*W*).

Le système peut être utilisé de trois manières différentes :

1. ***High***

Les combinaisons les moins probables ont les valeurs les plus élevées (valeur croissante dans le tableau 2).

2. ***Low***

Les combinaisons les plus probables ont les valeurs les plus élevées (valeur décroissante dans le tableau 2).

3. ***High / Low***

Il y a deux combinaisons gagnantes par manche une High et une Low et l'enjeu est partagé.

1.1.4 Échantillon

Nous appellerons échantillon les cartes que le joueur a à sa disposition pour former la meilleure combinaison de 5 cartes. Ces cartes peuvent être **communes**, c'est-à-dire qu'elles sont utilisables par l'ensemble des joueurs ou **privative**, utilisable par un seul joueur. Les cartes privatives peuvent elles-mêmes être soit fermées, uniquement visibles par le joueur, ou ouvertes et donc visibles de tous.

De ces caractéristiques découlent les trois grandes familles de poker :

1. ***Draw poker***

En français poker fermé. Chaque joueur reçoit cinq quartes toutes privatives et fermées.

2. ***Community card poker***

Une partie des cartes est commune et une autre est privative et fermée.

3. Stud poker

Toutes les cartes sont privatives mais une partie est ouverte et l'autres fermées.

Le nombre de cartes à la disposition du joueur varie d'un poker à l'autre. Dans le cas où le nombre de cartes composant l'échantillon est de plus de 5, la meilleure combinaison de 5 cartes sera prise en compte.

1.1.5 Ressources

Les ressources d'un joueur sont matérialisées par des jetons. Le nombre de jetons avec lequel un joueur peut débuter une partie est déterminé à l'avance et s'appelle la cave.

Les ressources déterminent également la capacité d'actions d'un joueur.

Au poker l'ensemble des jetons d'un joueur s'appelle le tapis. La règle du all-in (uniquement en no-limit) garantit au joueur de toujours pouvoir suivre une mise et cela même si ces ressources sont inférieures à la mise en question. Le joueur est alors déclaré all-in et ne concoure qu'à hauteur de sa mise.

1.1.6 Dessein

L'objectif premier au poker est de maximiser son nombre de jetons. Il existe deux moyens de gagner une manche de poker :

1. la première consiste à faire passer tous ces adversaires,
2. la seconde consiste à présenter lors de l'abattage une meilleure combinaison que les autres joueurs (en cas d'égalité on peut prétendre à une partie de l'enjeu)

1.1.7 Contraintes

Avant toute donne, les joueurs doivent placer les mises forcées qui sont de plusieurs sortes : les antes, les blindes, les options et le bring-in.

1.1.8 Processus

Le processus détermine la succession de phases qui caractérisent une manche.

Le battage (mélange des cartes) peut être considéré comme l'instant 0 d'une manche, les mises obligatoires sont alors placées par les joueurs.

Une donne correspond à une phase de distribution de cartes, une manche en comporte plusieurs.

Les enchères sont une phase successive à une donne pendant laquelle chaque joueur, toujours en lice, effectue une ou plusieurs actions.

1.1.9 Enjeu

L'enjeu représente l'ensemble des jetons qui ont été engagés par les joueurs. Avant le premier tour d'enchères, l'enjeu est constitué uniquement des mises obligatoires. Au poker l'enjeu est appelé le pot.

1.1.10 Alternatives

Nous appellerons alternatives l'ensemble des actions à disposition du joueur lors d'un tour d'enchères.

Ces actions sont :

- suivre (CALL)
- passer (FOLD)
- parole (CHECK)
- miser (BET)
- relancer (RAISE)

Quand les mises ne sont pas fixes (fixed limit), les deux dernières actions (*BET* & *RAISE*) impliquent l'annonce du montant de la mise. Ce montant est fixé par le joueur mais à l'intérieur de la plage déterminée par la règle en vigueur.

Les règles régissant les mises sont :

- No limit, un joueur peut miser n'importe quel montant de jetons, à hauteur de son tapis.
- Pot limit, un joueur peut miser n'importe quel montant de jetons, au maximum à hauteur du pot.
- Fixed limit, une limite de mise prédéterminée est appliquée à chaque tour d'enchères.
- Cap limit, une limite de mise prédéterminée est appliquée à la manche entière, une fois celle-ci atteinte plus aucune mise n'est possible et les donnes se succède jusqu'à l'abattage.

1.1.11 Superstructure

Une partie de poker est un ensemble de manches.

À chaque nouvelle manche, la position du dealer, identifiée par le bouton (*B*), avance d'un joueur dans le sens horaire.

Dans certains cas on peut changer de jeu de poker au sein d'une même partie.

1.2 Variante

La variante que nous allons utiliser est le Texas hold'em qui est sans conteste la variante la plus populaire aujourd'hui.

Sur bases des caractéristiques que nous venons de voir on peut classer cette variante comme suit :

- Domaine : Game
- Classe : Card game
- Genre : Poker
- Famille : Community card poker
- Variante : Texas hold'em
- Formes : No limit, Pot limit, Fixed limit, Cap limit
- Types : Full ring, heads-up, Short Handed
- Modes : Tournois ou cash game

1.2.1 Règles

Les actions sont menées dans le sens horaire.

Chaque joueur reçoit deux cartes fermées. Cinq cartes communes sont distribuées au fil des donnes. Le joueur forme donc la meilleure combinaison de cinq cartes à partir des 7 cartes (2 fermées + 5 communes) à sa disposition. Ceci se réalise en associant aucune, une ou deux des cartes fermées aux cartes communes.

En générale, les mises obligatoires sont au nombre de deux et sont appelées small blind (*sb*) et big blind (*bb*). La *sb* est placée par le joueur situé à gauche du dealer et la *bb* par le joueur suivant. La *bb* vaut le double de la *sb*.

Dans le cadre du no-limit, ou les mises ne sont pas fixes, une mise doit toujours être au minimum égale au montant de la bb. Si une mise à déjà été faite, la relance doit toujours être au moins du double du montant de la mise. En cas de sur relance il faut au moins doubler la valeur net de la relance précédente.

Une manche se déroule en maximum 4 phases, appelées respectivement « pré flop », « flop », « turn » et « river ».

Dans le cadre du limit, les mises sont fixées à l'avance. On appelle une mise dans les phases pré flop et flop, le small bet (*SB*). Dans les phases turn et river il s'agit du big bet (*BB*). En général, un *BB* équivaut à deux *SB*.

1.2.2 Déroulement d'une manche.

Phase 1 : « pré flop »

- Battage : Les joueurs participant au coup sont identifiés, les cartes sont mélangées et les mises obligatoires sont placées.
- Donne 1 : Le joueur identifié comme le dealer distribue les cartes une à une en commençant par le joueur situé immédiatement à sa gauche. Chaque joueur reçoit au total deux cartes privatives.
- Enchères 1 : Le premier tour d'enchères commence par le joueur situé immédiatement à gauche du dernier joueur ayant placé une mise obligatoire. En présence de mises, les seules actions possibles sont *FOLD*, *CALL* ou *RAISE*.

Phase 2 : « flop »

- Donne 2 : Trois cartes ouvertes sont distribuées, on appelle ces cartes le flop.
- Enchères 2 : Aucune mise obligatoire n'est présente, les actions possibles pour le premier joueur à parler sont *CHECK*, *BET* et dans une moindre mesure *FOLD*.

Phase 3 : « turn »

- Donne 3 : Une carte ouverte est distribuée, on appelle cette carte la turn.
- Enchères 3

Phase 4 : « river »

- Donne 4 : Une carte ouverte est distribuée, on appelle cette carte la river.
- Enchères 4
- Abattage : Si aucune mise n'a été faite, ou qu'au moins un joueur a suivi la mise ou la dernière relance on procède à l'abattage des cartes fermées en commençant par le joueur ayant misé le dernier. Le pot est attribué au joueur possédant la meilleure combinaison ou réparti en cas d'égalité.

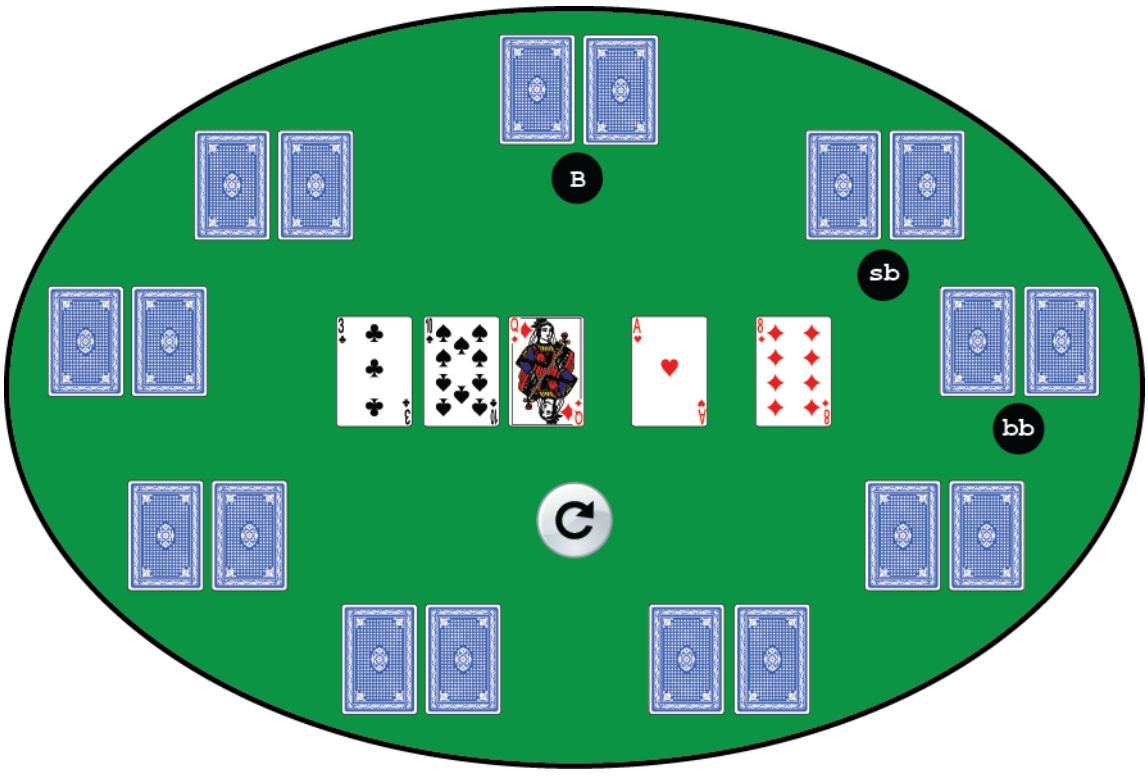

Figure 1 : Table de Texas hold'em (10 joueurs)

2 LES MATHÉMATIQUES

2.1 Combinatoire

Certains problèmes de calcul de probabilité peuvent se ramener à un calcul de dénombrement, en particulier ceux pour lesquels il y a un nombre fini d'issues possibles à l'expérience et où la probabilité de chaque issue est la même. Cette méthode consiste à compter (dénombrer) le nombre total de cas possibles et le nombre de cas favorables à la réalisation d'un évènement.

L'objet de la combinatoire s'appelle les complexions qui désignent les différents arrangements ou combinaisons d'un nombre fini d'éléments.

2.1.1 Principe multiplicatif

Si deux tâches indépendantes T_1 et T_2 doivent être exécutées l'une à la suite de l'autre, si la tâche T_1 peut être exécutée de n_1 façons différentes, et si la tâche T_2 peut être exécutée de n_2 façons différentes, alors la séquence T_1 T_2 peut être exécutée de $n_1 * n_2$ façons différentes.

2.1.2 Arrangements

Une complexion constituée de k éléments pris parmi n éléments différents en tenant compte de leur ordre est appelée un arrangement d'ordre k de n éléments.

$$A_n^k = \frac{n!}{(n-k)!}, \qquad 1 \leq k \leq n$$

Équation 1 : Nombre d'arrangements de n éléments sans répétition

$$^{r}A_n^k = n^k$$

Équation 2 : Nombre d'arrangements de n éléments avec répétition

2.1.3 Permutations

Un arrangement particulier ou n = k est appelé permutation.

$$P_n = n!$$

Équation 3 : Nombre de permutations de n éléments différents

$$P_n^{(n_1,n_2,\ldots,n_k)} = \frac{n!}{n_1!\ n_2!\ \ldots\ n_k!}, \qquad \sum_{i=1}^{k} n_i = n$$

Équation 4 : Nombre de permutations de n éléments avec des éléments égaux

2.1.4 Combinaisons

Une complexion constituée de k éléments pris parmi n éléments différents sans tenir compte de leur ordre est appelée une combinaison d'ordre k de n éléments.

$$C_n^k = \frac{n!}{k!\ (n-k)!}, \qquad 1 \leq k \leq n$$

Équation 5 : Nombre de combinaisons de n éléments sans répétition

$${}^{w}C_n^k = \frac{(n+k-1)!}{k!\ (n-1)!}$$

Équation 6 : Nombre de combinaisons de n éléments avec répétition

2.2 Probabilités

Les probabilités représentent un élément essentiel dans la pratique du poker. Les notions suivantes permettent de mieux appréhender l'architecture du jeu.

De manière générale, si n essais d'une expérience produit n_0 occurrences d'un évènement x, on définit la probabilité p(x) comme suit :

$$p(x) = \lim_{n\to\infty} \frac{n_0}{n}$$

Équation 7 : équation générale des probabilités

2.2.1 Définition

Définition axiomatique d'une probabilité : application de P qui associe un nombre réel p = P(A) à un évènement aléatoire A ∈ ε_1, telle que :

- **Axiome 1 :** P(A) est un nombre réel positif ou nul et ≤ 1 : 0 ≤ P(A) ≤ 1.
- **Axiome 2 :** la probabilité de l'évènement certain est 1 : P(**Ω**) = 1.
- **Axiome 3 :** la probabilité d'une somme d'évènements deux à deux incompatibles A_1, A_2, A_3, … est égale à la somme des probabilités des évènements : P(A_1 ∪ A_2 ∪ A_3 ∪ …) = P(A_1)+ P(A_2)+ P(A_3)+… .

2.2.2 Propriétés

- Pour la probabilité de l'évènement contraire $\bar{A}$ de A, on a $P(\bar{A}) = 1 - P(A)$.
- Pour la probabilité de l'évènement impossible Ø, on a $P(\emptyset) = 0$.
- Si A implique B (A ⊂ B), alors P(A) ≤ P(B).

2.2.3 Événements incompatibles

Deux événements sont dit incompatible si et seulement si leur réalisation simultanée est impossible.

$$p(A \cup B) = p(A) + p(B)$$

Exemple *: La probabilité qu'une carte d'un jeu de 52 cartes soit à la fois un quatre et une dame est nulle. Ces deux évènements sont donc incompatibles.*

2.2.4 Événements indépendants

On dit que des événements aléatoires sont indépendant si ils n'ont aucune influence l'un sur l'autre.

$$p(A \cap B) = p(A) * p(B)$$

Exemple *: La probabilité qu'une carte d'un jeu de 52 cartes soit un cœur n'influence pas la probabilité qu'elle soit un as. En effet la probabilité d'avoir un as dans un jeu de 52 carte est de 4/52 c'est-à-dire 1/13 et la probabilité d'avoir un as parmi les cœurs est de 1/13. Ces deux évènements sont donc indépendants.*

⇨ p(A ♥) = (4/52) * (13/52) = 1/52

2.2.1 Événements dépendants

On dit que des événements aléatoires sont dépendant si la réalisation de l'un affecte la réalisation de l'autre.

$$p(A \cap B) = p(A) * p(B \mid A)$$

Exemple *: Probabilité de tirer une paire d'As = p(AA) = (4/52) * (3/51) = 1/221*

2.2.2 Loi de probabilité

Une loi de probabilité est une donnée qui permet la caractérisation complète d'une variable aléatoire.

1. Espérance mathématique

L'espérance mathématique M(X) est une valeur numérique qui évalue le résultat moyen d'une expérience aléatoire.

$$M(X) = \sum_i x_i p_i$$

Équation 8 : Espérance mathématique d'une expérience aléatoire discrète

Dans le cadre d'un jeu tel que le poker, l'objectif est donc de maximiser l'espérance mathématique.

2. Variance

La variance V(X) représente la moyenne des carrés des écarts à la moyenne : elle permet de caractériser, tout comme l'écart type (σ), la dispersion des valeurs par rapport à l'espérance mathématique.

$$V(X) = \sum_i (x_i - m)^2 \, p_i = \sigma^2$$

Équation 9 : Variance d'une expérience aléatoire discrète

3. Loi normale

La loi normale est une loi de certaines variables aléatoires (valeurs mesurées), que l'on peut voir comme étant la loi d'une somme de grandeurs de contributions approximativement identiques et entachées d'erreurs.

$$f_n(X) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$

Équation 10 : Densité de probabilité

La loi normale standard est un cas particulier ou l'espérance est nulle (m=0) et l'écart type vaut un ($\sigma = 1$).

4. Théorème centrale limite

Représente la somme de n variables aléatoires indépendantes qui suivent une même loi qui s'approche de la loi normale lorsque n croit.

5. Maximum de vraisemblance

La méthode du maximum de vraisemblance consiste, en présence du résultat du tirage d'un échantillon, à se demander quelle était la probabilité de l'obtenir.

2.2.1 Inférence bayésienne[1]

On nomme inférence bayésienne la démarche logique permettant de calculer ou réviser la probabilité d'un événement. Cette démarche est régie en particulier par le théorème de Bayes. Dans la perspective bayésienne, une probabilité n'est pas interprétée comme le passage à la limite d'une fréquence, mais comme la simple traduction numérique d'un état de connaissance.

Une différence entre l'inférence bayésienne et les statistiques classiques, dites aussi fréquentistes, indiquée par Myron Tribus, est que :

- les méthodes bayésiennes utilisent des méthodes impersonnelles pour mettre à jour des probabilités personnelles, dites aussi subjectives (une probabilité est en fait toujours subjective, lorsqu'on analyse ses fondements),
- les méthodes statistiques utilisent des méthodes personnelles pour traiter des fréquences impersonnelles.

Les bayésiens font donc le choix de modéliser leurs attentes en début de processus (quitte à réviser ce premier jugement en donnant des poids de plus en plus faibles aux aprioris au fur et à mesure des observations), tandis que les

[1] Extrait de l'article Wikipédia sur l'Inférence bayésienne

statisticiens classiques se fixaient a priori une méthode et une hypothèse arbitraires et ne traitaient les données qu'ensuite.

2.3 Structure du jeu

2.3.1 Tableau

1. Cartes privatives

Il existe $C_{52}^{2} = 1.326$ mains de départ possible.

2. Cartes communes

Flop

Il existe $C_{52}^{3} = 22.100$ flop possible.

Turn

Il existe $C_{52}^{1} = 52$ turn possible.

River

Il existe $C_{52}^{1} = 52$ river possible.

2.3.2 Évaluations[5]

1. *Évaluation de la force*

Le calcul de la force brute d'une combinaison IHS (Immediate Hand Rank) est assez simple, il consiste à évaluer le nombre de victoires, de partages et de défaites contre l'ensemble de toutes les mains que l'adversaire peut avoir.

$$IHR = \frac{wins + \frac{ties}{2}}{wins + ties + losses}$$

Exemple :

Cartes privatives : Tc Jc

Flop : 2d Ts Kh

L'adversaire peut avoir $C_{47}^{2} = 1.081$ *cartes privatives possible.*

On gagne 899 fois, on partage 6 fois et on perd 176 fois.

⇨ $IHR = \frac{899 + 3}{899 + 6 + 176} = 83\%$

Cette solution suppose une distribution uniforme des cartes privatives de l'adversaire, hors cette hypothèse est assez éloignée de la réalité puisqu'un joueur va souvent jeter ses mauvaises mains pré flop.

Pour compenser ce problème, on peut pondérer l'ensemble des cartes privatives de l'adversaire grâce aux informations récoltées sur celui-ci. On utilise, en général, une table de pondération contenant toutes les cartes privatives possible, celle-ci est mise à jour à chaque fois que l'information est disponible. Le nouvel indice ainsi obtenu est appelé IHS (Immediate Hand Strength).

2. Évaluation du potentiel

La force ne tient compte que de la phase actuelle, hors au flop ou à la turn des cartes communes doivent encore arriver. Le potentiel représente soit la probabilité d'avoir la meilleure combinaison mais de perdre le coup lors de l'abattage (potentiel négatif) ou au contraire d'avoir une moins bonne combinaison mais de gagner la manche à l'abattage (potentiel positif).

$$Pot.positif = p(gagner \mid derrière) + \frac{p(partager \mid derrière)}{2} + \frac{p(gagner \mid partage)}{2}$$

$$Pot.négatif = p(perdre \mid devant) + \frac{p(partager \mid devant)}{2} + \frac{p(perdre \mid partage)}{2}$$

3. Combinaison force et potentiel

L'EHS (Effective Hand Strength) combine l'IHS (la force) et le potentiel.

$$EHS = IHS + (1 - IHS) \times PositivePotential$$

4. Cote du pot

La cote du pot représente le ratio entre le montant du *CALL* (ce que le joueur doit investir) et le pot qu'il peut gagner.

$$\text{Cote du pot} = \frac{montant\ à\ payer}{montant\ à\ payer + valeur\ du\ pot}$$

On parle de cote implicite quand on envisage les enchères suivantes et les montants que l'adversaire pourrait encore investir.

5. Espérance

L'espérance ou EV (Expected value) est un paramètre primordial dans l'évaluation de la pertinence d'une action.

***Exemple** : Nous sommes favoris à 60% dans une manche ou le pot total est de 100.*

$$EV = (0.6) \times (100) + (0.4) \times (-100) = 20$$

2.3.3 Analyse de la complexité[1]

Nous allons maintenant nous faire une idée de la complexité du poker Texas Hold'em limit Heads-Up en parcourant les différents facteurs entrant en compte.

1. ***Donnes***

Cartes privatives

Pré flop, 2 cartes privatives sont distribuées à chaque joueur.

Joueur 1 : $C_{52}^{2} = 1326$

Joueur 2 : $C_{50}^{2} = 1225$

⇨ **1.624.350**

Cartes communes

Au flop, 3 cartes communes sont distribuées.

Flop : $C_{48}^{3} = 17296$

⇨ **17.296**

À la turn, 1 carte commune est distribuée.

Flop : $C_{45}^{1} = 45$

⇨ **45**

À la river, 1 carte commune est distribuée.

Flop : $C_{44}^{1} = 44$

⇨ **44**

2. Enchères

Comme nous l'avons vu précédemment, les actions possibles lors d'un tour d'enchères sont :

- suivre (CALL)
- passer (FOLD)
- parole (CHECK)
- miser (BET)
- relancer (RAISE)

On peut considérer que *CALL* et *CHECK* sont similaires dans la mesure où ces deux actions consistent à payer le montant minimum afin de rester dans le coup.

BET et *RAISE* sont également similaires puisqu'elles visent chacune à augmenter le montant à payer pour rester dans le coup.

Après avoir appliqué ces deux abstractions, nous avons donc :

- "f" = *FOLD*
- "c" = *CALL* et *CHECK*
- "r" = *BET* et *RAISE*

Le règlement de la compétition limite le nombre de *BET*/*RAISE* à 3 pré flop et à 4 à partir du flop.

Les actions et les séquences d'actions ont différentes implications :

- "f" entraine la fin du mot et la fin de la manche,
- "c" prolonge le mot et donc la manche,
- "cc" et "rc" entrainent la fin du mot et la manche continue dans la phase suivante,
- "r" prolonge le mot et donc la manche.

Un maximum de 19 mots peuvent être composé à partir de l'alphabet {c, r, f} et des règles en vigueur.

1	2	3	4	5	6	Total
f	cf	crf	crrf	crrrf	crrrrf	
	cc	crc	crrc	crrrc	crrrrc	
	rf	rrf	rrrf	rrrrf		
	rc	rrc	rrrc	rrrrc		
1	4	4	4	4	2	**19**

Tableau 3 : Dictionnaire des mots autorisés sur l'alphabet (c, r, f)

Les mots se terminant par l'action "f" (soulignés dans le tableau ci-dessus) terminent la manche et ne nous intéressent donc pas dans l'estimation de la complexité.

Le mot "cf" (en rouge dans le tableau ci-dessus) suggère qu'un joueur qui a la possibilité de voir la donne suivante en réalisant un "c", préfère opter pour l'action "f" lui faisant immédiatement perdre la partie. Ce choix peut être considéré comme non rationnel. À la rivière, ce choix peut parfois se justifier pour dissimuler de l'information si le joueur n'a plus aucune chance de gagner le coup.

Le mot "f" (en bleu dans le tableau ci-dessus) suggère qu'un joueur qui a la possibilité de réaliser un "c", préfère opter pour l'action "f" lui faisant immédiatement perdre la partie. Ce choix peut être considéré comme non rationnel, sauf pré flop ou "c" représente un *CALL* et donc l'investissement de la différence entre la *sb* et la *bb* et également à la rivière pour les raisons exposées ci-dessus.

Pré flop

La limite du nombre de "r" étant de 3 pré flop, 4 mots (en vert dans le tableau ci-dessus) peuvent être soustraits au dictionnaire. Ils restent donc 7 mots possible : {cc, rc, crc, crrc, crrrc, rrc, rrrc}.

⇨ **7**

Flop et turn

La limite de "r" passe à 4 et 9 mots sont donc possible : {cc, rc, crc, crrc, crrrc, rrc, rrrc, rrrrc, crrrrc}.

⇨ **9**

River

La rivière étant obligatoirement une phase terminale, tous les mots doivent être pris en compte, soit 19 mots possible : {f, cf, crf, crrf, crrrf, crrrrf, cc, crc, crrc, crrrc, crrrrc, rf, rrf, rrrf, rrrrf, rc, rrc, rrrc, rrrrc}.

⇨ **19**

3. Complexité

Si l'on comptabilise toutes les combinaisons possibles on obtient :

$$1.624.350 \times 7 \times 17.296 \times 9 \times 45 \times 9 \times 44 \times 19 = 5{,}99 \times 10^{17}$$

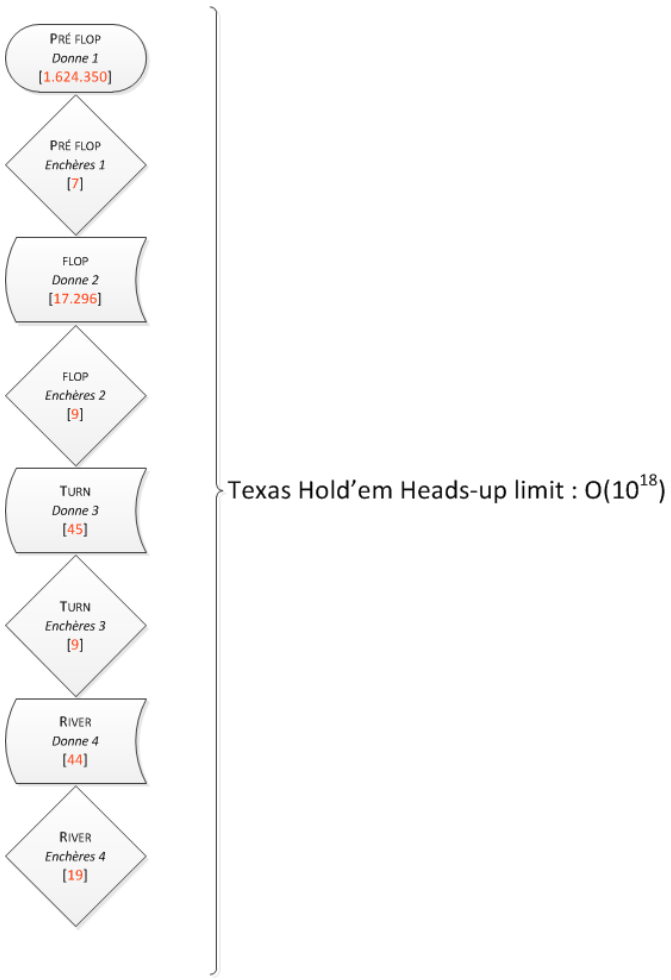

Figure 2 : Représentation de la complexité du Texas Hold'em Heads-up limit (tiré de [1])

3 LA THÉORIE DES JEUX

Que ce soit chez l'homme ou chez certains animaux les vertus du jeu sont bien connues, notamment son rôle positif dans l'apprentissage de tâches complexes.

Si l'on fait abstraction de son caractère ludique, on peut généraliser le jeu à quasi toutes les interactions entre les personnes. En effet, que ce soit lors d'une vente aux enchères, une négociation de contrat, une guerre ou simplement un déplacement en voiture dans la circulation on est d'une certaine manière dans un jeu.

C'est sans doute ces multitudes d'applications qui font de la théorie des jeux l'une des disciplines les plus prometteuses. Pour preuve de l'engouement qu'elle rencontre, les trois « Prix Nobel d'Économie » décernés à des travaux sur le sujet lors des vingt dernières années (1994, 2007 et 2012). Mais l'économie n'est pas le seul domaine à tirer parti de la théorie des jeux, elle est également largement utilisée en biologie, en politique, en droit, en philosophie et bien entendu dans les sciences de l'informatique.

L'intérêt de la théorie des jeux pour le poker remonte à ses prémices puisqu'aussi bien Émile Borel[2] que John von Neumann[13] l'ont étudié dans leur livre respectif, ouvrages que l'on peut considérer comme la genèse de la discipline. Ce choix n'est pas dû au hasard mais bien au caractéristiques du poker que nous allons décrire ci-après tout en présentant certains grands principes de la théorie de jeux.

3.1 Caractéristiques des jeux

Pour Peter Morris[18], un jeu implique la présence des éléments suivants :

- Un ensemble fini de joueurs,
- Un ensemble fini d'actions que chaque joueur peut prendre à des moments spécifiques du jeu,
- Un profit sous forme numérique assigné à chaque joueur à la fin du jeu. Ce profit pouvant être une valeur négative (une perte), une valeur positive (un gain) ou 0 (une égalité).

En plus de ces éléments un jeu peut également contenir :

- Des évènements aléatoires comme le tirage d'une carte dans un paquet,
- Des informations cachées, c'est-à-dire des informations non disponibles pour un joueur lors de sa prise de décision.

Nous allons maintenant caractérisés plus avant les jeux en nous intéressant à leurs propriétés intrinsèques.

3.1.1 Jeux à somme nulle / non-nulle

Quand dans un jeu à deux joueurs l'intérêt d'un agent est strictement opposé à celui de l'autre, c'est-à-dire que les agents ont des préférences strictement opposées, on dit que le jeu est à somme nulle. On parle également dans ce cas de jeu strictement compétitif.

3.1.2 Jeux à information complète / incomplète

Les informations potentiellement accessibles par un agent lors d'une prise de décision sont :

- Les différentes actions qui lui sont autorisées,
- Les différentes actions autorisées aux autres agents,
- Le profit résultant de ces actions,

- Les motivations des autres agents.

Si l'une des informations n'est pas connue de l'agent lors de sa prise de décision, on parle de jeu à information incomplète.

3.1.3 Jeux à information parfaite / imparfaite

Un jeu à information incomplète est par définition un jeu à information imparfaite. Les jeux à information complète peuvent être à information imparfaite si ils sont simultanés (voir ci-dessous) soit si des évènements aléatoires ne sont pas connu de l'agent lors de la prise de décision.

3.1.4 Jeux coopératifs / non-coopératifs

Un jeu coopératif se caractérise par la possibilité que les joueurs ont de se rassembler en coalition afin d'améliorer le bénéfice de chaque membre de cette coalition.

3.1.5 Jeux à 2 joueurs / n joueurs

Le nombre de joueurs est un élément sensible de la complexité des jeux. On fait la distinction entre les jeux à 2 joueurs et à plus de deux joueurs.

3.1.6 Jeux à mémoire parfaite / imparfaite

Par mémoire parfaite, on entend que l'agent est capable de retenir l'ensemble des évènements qui se sont déroulés depuis le début du jeu. Les jeux à information imparfaite supposent une amnésie au moins partielle de la part des joueurs.

3.1.7 Jeux simultanés / séquentiels

On dit d'un jeu qu'il est de type simultané si les agents doivent choisir en même temps l'action qu'ils vont prendre, à l'inverse quand les agents agissent chacun à leur tour, en connaissance des éventuelles actions de leur(s) adversaire(s), le jeu est de type séquentiel.

3.2 Utilité

L'une des modalités de la théorie des jeux est que les agents agissent de manière rationnelle, c'est-à-dire que leurs choix sont fondés sur la raison et visent à atteindre l'état qui leur convient le mieux.

L'utilité peut être vue comme la mesure subjective de la satisfaction de l'agent.

3.2.1 Fonction d'utilité

Une fonction d'utilité au sens définit par von Neumann et Morgenstern équivaut à :

$$u_i : \Omega \rightarrow \mathbb{R}$$

Ou Ω représente l'ensemble des résultats finaux du jeu.

3.3 Représentations

3.3.1 Forme normale

L'une des manières courantes pour représenter un jeu est connue sous le nom de forme normale alias forme stratégique. On définit le jeu par :

- $N = \{1, \dots, n\}$, l'ensemble des joueurs,
- S_i, l'ensemble non vide des actions ou stratégies pures du joueur i
- $u_i : S_1 \times \dots \times S_n \to \mathbb{R}$, la fonction d'utilité, de gain ou de paiement du joueur i ou $S_1 \times \dots \times S_n = S$ le produit cartésien des stratégies des différents joueurs.

Matrice de gain

Dans un jeu à deux joueurs avec un ensemble fini de stratégies pour chacun des deux joueurs, il est courant de représenter le jeu sous sa forme normale à l'aide d'une matrice des gains ou matrice des paiements.

Par exemple, pour le jeu « Pierre-feuille-ciseaux » la matrice des gains est :

		Joueur 2		
		P	**F**	**C**
Joueur 1	**P**	0,0	-1,1	1,-1
	F	1,-1	0,0	-1,1
	C	-1,1	1,-1	0,0

Tableau 4 : Le jeu « Pierre-feuille-ciseaux » représenté sous forme d'une matrice.

3.3.2 Forme extensive

Les jeux peuvent être également représentés par un arbre.

Un jeu sous forme extensive est défini par :

- un ensemble $N = \{1, \dots, n\}$ de joueurs
- un arbre fini composé de :
    - un ensemble de nœuds $\{A,\ B,\ C, \dots\}$ représentant les coups
    - un ensemble de branches $\{x,\ y,\ z, \dots\}$ représentant les alternatives à chaque coup
- une fonction de nommage qui indique à chaque nœud quel est le joueur qui doit jouer
- une fonction d'utilité, de gain ou de paiement qui associe à chaque nœud terminal un vecteur de nombres représentant les gains de chacun des joueurs
- une partition des nœuds en un ensemble d'ensembles d'informations représentant les croyances (imparfaites) des joueurs

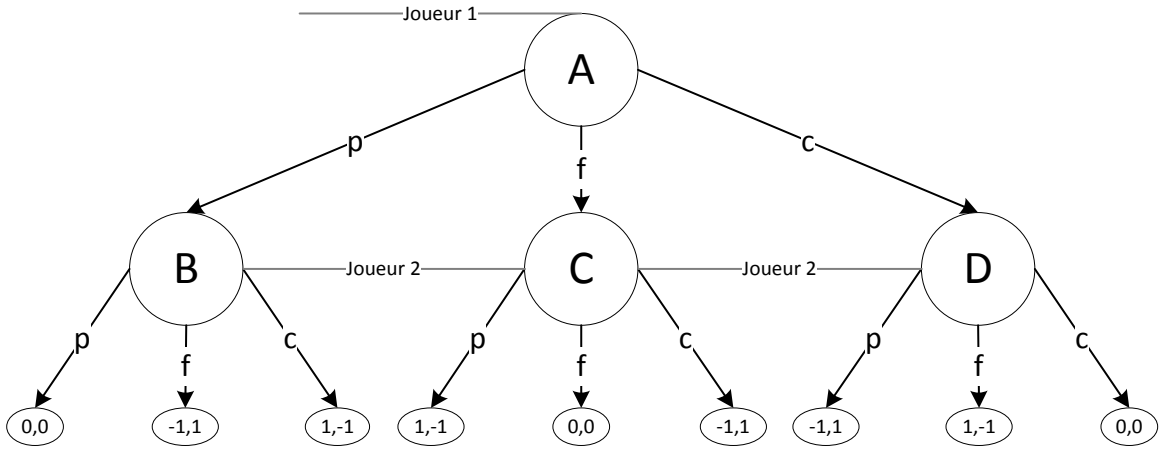

Figure 3 : Le jeu « Pierre-feuille-ciseaux » représenté sous forme extensive.

3.4 Stratégies

La stratégie peut se définir comme l'application entre les différents états d'un jeu et les actions qu'un agent effectuera dans ces états.

Une stratégie est dite statique si elle n'évolue pas en cours de jeu et au contraire « adaptive » si elle évolue dans le temps.

3.4.1 Stratégie pure

Une stratégie pure du joueur i est une action ou un plan d'actions qui prescrit une action de ce joueur pour chaque fois qu'il est susceptible de jouer. On note par S_i l'ensemble des stratégies pures du joueur i et par s_i une stratégie pure de ce joueur.

3.4.2 Stratégie mixte

La notion de stratégie pure est étendue à celle de stratégie mixte qui est définie comme une distribution de probabilité sur l'ensemble des stratégies pures.

Dans le cadre du poker une stratégie est déterminée par un triplet [f, c, r] de probabilités pour chaque état du jeu ou f,c et r sont respectivement les probabilités de *FOLD*, *CALL* ou *RAISE*.

3.4.3 Stratégies dominantes

On parle de stratégie dominante quand pour un joueur une stratégie apparait meilleure qu'une autre et ce quelle que soit la stratégie de l'adversaire.

En d'autres mots, dans un jeu à information parfaite, pour deux stratégies A et B possible pour un joueur, B domine A si le gain associé à B est supérieur ou égal à celui associé à A pour toute stratégie de l'adversaire. Si le gain est supérieur ou égale on parle de stratégie faiblement dominante, s'il est strictement supérieur on dit que la stratégie est strictement dominante.

Dans un jeu en forme stratégique, on dit qu'une stratégie $s'_i \in S_i$ est dominante pour le joueur i si, quel que soit $\hat{s}_i \in S_i$ et $\hat{s}_i \neq s'_i$, les inégalités :

$$u_i(s'_i, s_{-i}) \geq u_i(\hat{s}_i, s_{-i})$$

sont satisfaites pour tout $s_{-i} \in S_{-i}$.

Si dans un jeu donné tous les joueurs disposent d'une stratégie dominante et qu'ils choisissent effectivement cette stratégie, le résultat du jeu est appelé équilibre en stratégies dominantes.

3.4.4 Meilleure réponse

Une ou plusieurs stratégies sont qualifiées de meilleures réponses quand elles maximisent la fonction d'utilité d'un joueur donné par rapport aux stratégies de ses adversaires.

Une stratégie s'_i est une meilleure réponse du joueur i à un profil de stratégies $s_{-i} \in S_{-i}$ de l'autre joueur si :

$$u_i(s'_i, s_{-i}) \geq u_i(\hat{s}_i, s_{-i})$$

pour tout $\hat{s}_i$.

3.4.5 Équilibre de Nash

Le concept d'équilibre de Nash (John Nash, 1950) décrit un équilibre dans un jeu non coopératif ou aucun joueur ne peut modifier seul sa stratégie sans affaiblir sa position.

Un profil de stratégies $s \in S$ est un équilibre de Nash si s'_i est une meilleure réponse à s_{-i} pour tout i. Autrement dit, s est un équilibre de Nash si :

$$u_i(s'_i, s_{-i}) \geq u_i(\hat{s}_i, s_{-i})$$

pour tout i et $\hat{s}_i$.

Epsilon-équilibre

Abrévié ε-Nash un epsilon-équilibre est obtenu en appliquant des abstractions simplificatrices au jeu. « ε » représente le montant maximal qu'un joueur peut gagner en déviant de sa stratégie.

3.5 Jeu d'exploitation

Le jeu d'exploitation ou « exploitative play » en anglais est une technique qui consiste à tirer avantage de toutes les informations disponibles, en ce compris les faiblesses de l'adversaire, afin de maximiser l'espérance mathématique.

Cela nécessite l'élaboration d'un modèle de l'adversaire afin de « prévoir » ses futures actions.

Les stratégies d'exploitation peuvent être statique ou « adaptive ».

Ces stratégies dévient, par essence, d'un équilibre de Nash et sont donc elles-mêmes exploitables.

3.6 Analyse du poker

Comme nous l'avons déjà dit, le poker possède une série de caractéristiques qui en font un sujet d'étude très intéressant pour les chercheurs de différentes disciplines. Certaines des propriétés marquantes du jeu sont décrites ci-dessous.

Les cartes privatives des adversaires étant par définition inconnues du joueur le jeu est dit à **information imparfaite**. L'incertitude générée par cette caractéristique induit la nécessité de recourir à la tromperie mais également à s'adapter à la tromperie de l'adversaire. Deux types de tromperie sont à considérer :

- Le bluff, qui se produit lorsqu'un joueur pensant avoir une mauvaise main cherche à faire croire aux autres joueurs que la main est bonne.
- La neutralisation, qui se produit lorsqu'un joueur pensant avoir une bonne main cherche à faire croire aux autres joueurs que la main est mauvaise.

Cette caractéristique avantage le joueur variant son style de jeu par l'utilisation de stratégies mixtes.

La distribution aléatoire des cartes à différentes étapes du jeu introduit un **aspect stochastique des profits**. Cette variance importante dans les résultats rend l'évaluation des performances difficile. La **gestion du risque** par l'analyse des stratégies et de leurs conséquences est dès lors complexe mais primordiale.

Au poker, un joueur peut remporter la partie si tous les autres joueurs *FOLD*, avec pour conséquence qu'aucune information sur les cartes fermées des joueurs n'est alors disponible. Les **états cachés sont donc partiellement observables**, ce qui rend l'élaboration d'un **modèle de l'adversaire** et son exploitation beaucoup plus compliquée.

Le nombre de joueur varie de 2 à 10 ce qui classe le poker aussi bien dans la catégorie des jeux à deux joueurs que dans celle multi-joueurs. C'est dernier

sont, malgré l'absence de coalition au poker (jeu non-coopératif), nettement plus complexe.

Les jeux rassemblant ces caractéristiques sont parmi les problèmes les plus compliqués en informatique théorique.

La variante du poker à laquelle nous allons nous intéresser est le Texas Hold'em heads up limit qui appartient à la famille des **jeux à deux joueurs à somme nulle**.

4 L'INTELLIGENCE ARTIFICIELLE[20]

4.1 Estimateurs de performances

4.1.1 Small bets per hand (sb/h)

Il s'agit du nombre de « small bets » en limit ou « big blinds » en no limit gagner ou perdu divisés par le nombre total de manches jouées. Dans la pratique on utilise souvent le millième de « small bets ».

4.1.2 Exploitabilité ε

ε peut être estimé en calculant la « meilleure réponse » à la stratégie d'un agent. En pratique, pour une stratégie statique connue, on définit à chaque état du jeu, l'action qui maximise l'espérance. Par état du jeu on entend ici les états distinguables.

4.1.3 « Annual Computer Poker Competition » (ACPC)

L'ACPC est la compétition de référence du domaine. Elle permet donc d'estimer les performances d'un agent par rapport à ce qui se fait de mieux.

4.2 Algorithmes

Nous abordons ici des algorithmes de base qui permettront une meilleure compréhension de notions qui seront énoncées par la suite.

4.2.1 Minimax

L'algorithme minimax est un algorithme qui s'applique aux jeux à deux joueurs à somme nulle et à information complète (ex. les échecs).

Cet algorithme passe en revue toutes les possibilités pour un nombre limité de coups et leur assigne une valeur qui prend en compte les bénéfices pour le joueur et pour son opposant. Le meilleur choix est alors celui qui minimise les pertes du joueur (pire des cas) tout en supposant que l'opposant cherche au contraire à les maximiser.

Principe :

L'arbre de jeu est visité afin de faire remonter à la racine une valeur. Cette valeur est appelée « valeur de jeu » et est calculé récursivement de la manière suivante :

Soit *p* un nœud de l'arbre de jeu et *f* est une fonction d'évaluation de la position du jeu. Alors :

$$minimax(p) = f(p) \text{ si } p \text{ est une feuille de l'arbre}$$

$$minimax(p) = MAX(minimax(O_1), \dots, minimax(O_n)) \text{ si } p \text{ est un nœud Joueur avec fils } O_1, \dots, O_n$$

$$minimax(p) = MIN(minimax(O_1), \dots, minimax(O_n)) \text{ si } p \text{ est un nœud Opposant avec fils } O_1, \dots, O_n$$

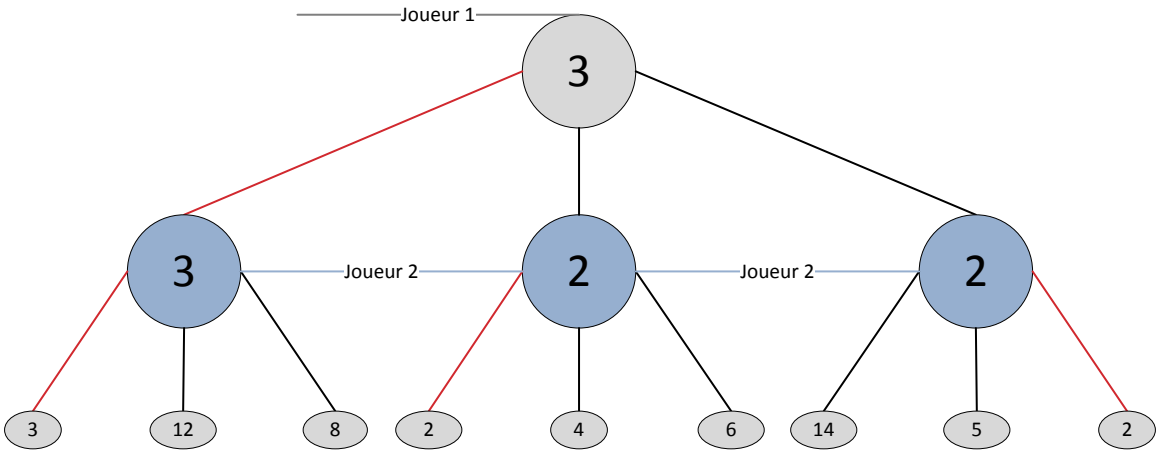

Figure 4 : Application de Minimax sur un arbre de jeu

4.2.2 Negamax

Negamax n'est pas à proprement parler un nouvel algorithme, il s'agit en réalité d'une implémentation différente de minimax.

Partant du principe que $max(a,b) == -min(-a,-b)$, on peut, plutôt que d'implémenter deux sous-routines différentes (MAX et MIN), n'utiliser que MAX et utiliser les valeurs opposées (négatives) aux valeurs qui devraient être passées à MIN.

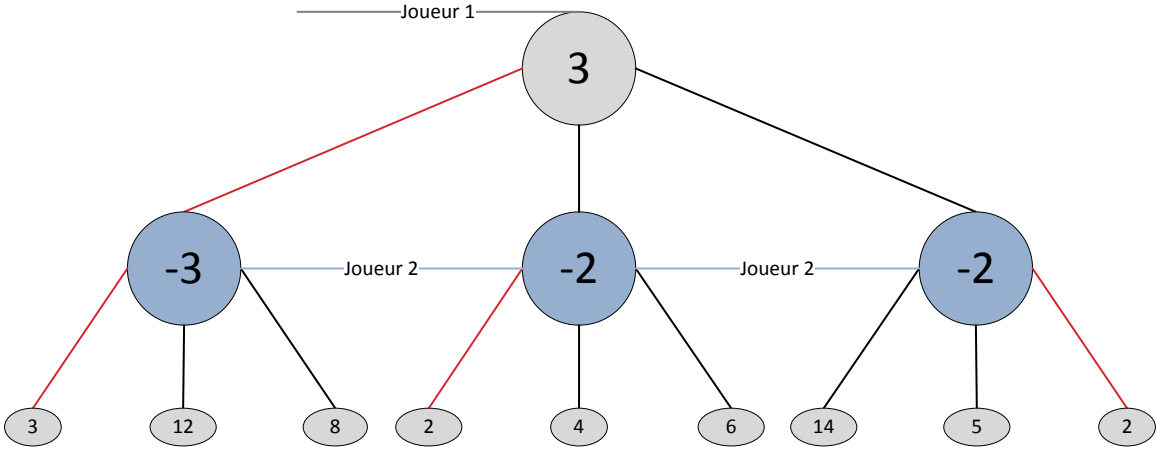

Figure 5 : Application de Minimax sur un arbre de jeu

4.2.3 Élagage alpha-bêta

L'élagage alpha-bêta (Alpha–beta pruning en anglais) est une technique d'optimisation applicable à minimax.

Dans l'exploration minimax, le nombre d'états à examiner dépend exponentiellement de la profondeur de l'arbre. La technique d'élagage alpha-bêta permet de calculer la décision minimax appropriée sans examiner tous les nœuds de l'arbre de jeu.

Principe :

Considérons un nœud *n* de l'arbre (figure x) tel que le joueur a le choix de se déplacer jusqu'à ce nœud. Si le joueur dispose d'un meilleur choix *m* au niveau du nœud parent, voire plus haut, alors *n* ne sera jamais atteint dans le jeu et on peut donc l'élaguer.

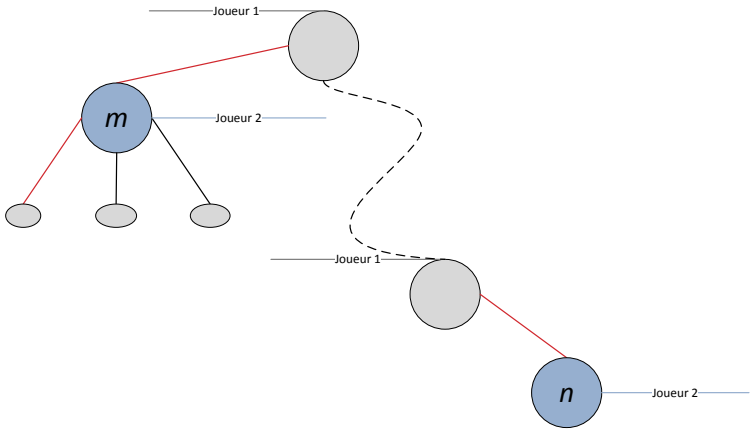

Figure 6 : Représentation d'un élagage alpha-bêta

L'élagage alpha-bêta tire son nom des deux bornes qui sont remontées le long du chemin :

- α : la valeur du meilleur choix provisoire pour MAX
- β : la valeur du meilleur choix provisoire pour MIN

L'exploration met à jour les valeurs de α et de β à mesure qu'elle progresse et élague les branches restantes d'un nœud.

L'efficacité de cette technique est sensiblement dépendante de l'ordre dans lequel les états sont examinés. Elle peut résoudre un arbre à peu près deux fois plus profondément que minimax pendant la même durée.

4.2.4 Simplex

L'algorithme du simplexe est un algorithme de résolution des problèmes d'optimisation linéaire.

Le nom de l'algorithme est dérivé de la notion de simplexe mais en réalité, l'algorithme n'utilise pas de simplexes, mais certaines interprétations de l'ensemble admissible du problème renvoient au concept de simplexe.

Principe :

Si un problème de programmation linéaire en forme standard possède une solution optimale, alors il existe une solution de base admissible qui soit optimale. La méthode du simplexe consiste à passer d'une solution de base admissible à l'autre, en réduisant le coût.

Le problème en forme standard s'écrit :

$$(P_L)\begin{cases} inf_x C^T x \\ Ax = b \\ x \geq 0. \end{cases}$$

Avec :

- $A \in \mathbb{R}^{m \times n}$ une matrice réelle de type $m \times n$ et $b \in \mathbb{R}^m$.
- lignes de A linéairement indépendantes
- l'ensemble admissible se note $A_p \coloneqq \{x \in \mathbb{R}^n : Ax = b, x \geq 0\}$

L'algorithme part d'une solution de base admissible et recherche une solution de base admissible voisine qui améliore l'objectif. Une fois qu'il n'en trouve plus on est en présence d'un minimum local, hors en programmation linéaire un minimum local équivaut à un minimum global.

4.3 Abstractions

Nous avons vu que la complexité du poker (Texas Hold'em limit Heads-up) était d'un peu moins que $O(10^{18})$. Celle-ci est bien trop importante pour pouvoir appliquer les techniques de calcul classiques. Les différentes formes d'abstractions que nous allons énoncer ici ont toutes pour but de diminuer la complexité du jeu, certaines provoquent des pertes d'informations et d'autres au contraire sont transparentes.

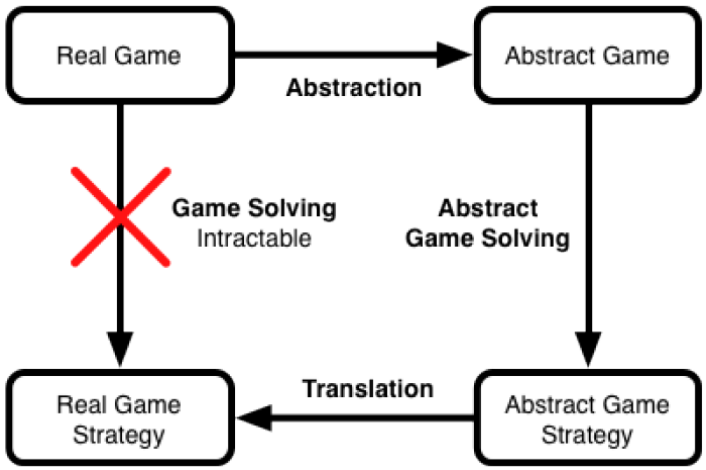

Figure 7 : Schéma d'utilisation des abstractions (tiré de *)

4.3.1 Sans perte

Comme nous l'avons vu précédemment les différentes combinaisons de cartes privatives pour le joueur 1 sont au nombre de 1326 mais certaines ne présentent entre elles que des différences non stratégiques dans la phase pré flop.

On peut donc réaliser une série d'abstractions tels que :

$[Ah, As] = [Ad, As] = [Ac, As] = [Ad, Ac] = [Ah, Ac] = [Ad, Ah]$

⇨ AA

[Ah, Ks] = [Ad, Ks] = [Ac, Ks] = [Ad, Kc] = [Ah, Kc] = [Ad, Kh] = [As, Kh] = [As, Kd] = [As, Kc] = [Ac, Kd] = [Ac, Kh] = [Ah, Kd]

⇨ *AK*

[Ah, Kh] = [Ad, Kd] = [Ac, Kc] = [As, Ks]

⇨ *AKs*

Avec l'étiquette "s" pour suited (assorti).

Si l'on applique ces abstractions on passe de 1326 à 169 combinaisons de cartes privatives, soit un rapport de quasi 8.

Ce type d'abstractions n'est malheureusement pas suffisant pour atteindre une taille qui permettrait de résoudre le jeu, elle a également pour désavantage de ne convenir qu'à une phase. En effet, l'exemple ci-dessus fonctionne très bien pré flop, mais lors de la transition vers le flop les information concernant l'enseigne sont nécessaires.

4.3.2 Avec pertes

Afin de réduire encore la complexité, on utilise des techniques qui vont affecter l'intégrité stratégique du jeu et qui ne garantissent donc plus une solution optimale, on dit qu'elles sont pseudo-optimal.

1. ***Bucketing***

Le bucketing est une technique qui consiste à répartir des combinaisons, partageant des similarités stratégiques, dans des classes d'équivalence.

Expectation-based abstraction

Cette méthode, classe les combinaisons selon leur probabilité de gagner à l'abattage appelé E[HS] (Expected Hand Strength).

Exemple *: Bucketing en 5 classes basées sur l'E[HS]*

[0.0–0.2][0.2–0.4][0.4–0.6][0.6–0.8][0.8–1.0]

On modélise alors le jeu en utilisant les classes plutôt que les cartes.

Le même travail est réalisé pour les différentes phases du jeu, et il faut alors déterminer les probabilités de transitions entre les classes de ces phases.

Potential-aware automated abstraction

Cette méthode, développée par Gilpin et al.[8], utilise une approche multidimensionnelle pour la distribution des combinaisons dans les différentes classes.

Le bucketing commence par la première phase : pré flop. Le nombre de classes pour chaque phase est déterminé à l'avance. Afin de capturer le potentiel d'une combinaison il faut préalablement réaliser un premier bucketing des phases suivantes. On travaille de bas en haut en commençant donc par la river qui possède la particularité d'être exclusivement terminal, le potentiel n'a donc plus d'importance et seul la force réel de la combinaison compte. On utilise l'algorithme des k-moyennes (k-means clustering) afin de répartir les combinaisons dans les classes. Ensuite, pour chaque combinaison de la turn, on calcule la probabilité d'évoluer vers chaque classe de la river. Dotées de leurs nouvelles caractéristiques les combinaisons de la turn sont elle- même réparties dans des classes via l'algorithme des k-moyennes. On recommence ce processus jusqu'à obtenir les classes pré flop. Pour les autres phases on recommence toute la procédure à la différence qu'on ne considère que les combinaisons d'une classe de la phase précédente. Par exemple, pour le flop on commencera par appliquer la procédure seulement aux combinaisons contenue dans la première classe, ensuite de la seconde, etc.

2. Mémoire imparfaite[22]

A priori il nous parait normal qu'un agent impliqué dans un jeu mémorise parfaitement toutes ses observations afin de prendre les meilleures décisions. Faire abstraction d'une partie plus ou moins grande de ces informations fais même disparaître la garantie que la solution soit optimale.

Néanmoins relâcher cette contrainte dans le cas de jeux présentant une très grande complexité s'avère être une solution performante qui permet d'allouer

plus de ressources au calcul de la phase en cours au détriment des précédentes.

3. Limite des enchères

La limite des enchères (Betting round reduction) consiste à limiter le nombre d'actions possibles. Le règlement de l'ACPC prévoit un maximum de 3 *BET/RAISE* dans la phase pré flop.

4. Élimination de phase

Une solution plus radicale consiste à éliminer des phases entières du jeu. Par exemple, un jeu qui se terminerait à la turn ou un jeu qui ne permettrait des enchères que pré flop, seul resterait les donnes des phases flop, turn et river.

4.4 Techniques[20]

4.4.1 Base de connaissance

Les techniques utilisant une base de connaissance nécessitent l'aide d'un expert dans le domaine envisagé.

1. Base de règles

Une base de règle, en anglais rule-base, est une collection de règles de types si/alors (voir figure ci-dessous) pour différents scénarios susceptibles de se produire.

C'est sans aucun doute la manière la plus naturelle d'aborder le problème et c'est d'ailleurs celle qui a souvent été utilisée dans les premiers agents. Malheureusement étant donné la complexité du jeu, cette technique s'est rapidement avérée beaucoup trop lourde et incapable de produire des agents de haute performance.

```
Action preflopAction(Hand hand, GameState state){
 if( state.betsToCall > 2 &&
     state.opponentsInHand > 1 &&
     state.relativePosition > 0.8 &&
     hand.AAo){
       return getAction(new Triple(0.0, 0.05, 0.95));
 } else if...
}
```

Figure 8 : Exemple de code d'un programme rule-based

2. Base de formules

Une base de formules, en anglais formula base, est une forme généralisée de la base de règles. Le système accepte en entré une collection de données, pouvant posséder des poids différents, décrivant des informations essentielles

concernant l'état du jeu. Il s'agit dans le cas qui nous intéresse des représentations numériques (voir structure du jeu) de la force des combinaisons (mains) et de la cote du pot.

- Immediate hand strength (IHS)
- Hand potential
- Effective hand strength (EHS)
- Pot odds

Ces informations sont alors prises en compte, et une action pour l'état en cours est passée en sortie.

4.4.2 Simulation

Une autre approche du problème est de simuler dynamiquement le jeu afin de faire ressortir les meilleures stratégies.

L'une des manières de procéder, consiste à parcourir exhaustivement l'arbre de jeu, quand une feuille est atteinte on applique une fonction d'utilité et on remonte la valeur obtenue au nœud intermédiaire. C'est le cas pour l'algorithme minimax avec élagage alpha-beta utilisé avec succès dans les jeux à information parfaite.

Son application aux jeux à information imparfaite est par contre problématique, en effet le manque d'information empêche de connaitre l'état exact dans lequel on se trouve.

Une méthode convient bien à la modélisation de problèmes complexes comportant une part d'incertitude, la méthode de Monte Carlo. Il s'agit en réalité d'un ensemble de méthodes qui ont en commun d'estimer une valeur numérique en utilisant des techniques probabilistes. Plus concrètement, en simulant aléatoirement les évènements probables un grand nombre de fois, on finit par converger vers des valeurs représentatives de la valeur réel recherchées (par exemple la valeur de π, une surface, un volume, ...).

L'une des techniques appliquant cette méthode est appelée MCTS (Monte Carlo tree search), celle-ci donne d'excellents résultats notamment au backgammon et au Go.

1. Monte-Carlo Tree Search (MCTS)

MCTS est donc une technique de simulation ayant pour objectif la recherche de décisions optimales. Elle combine la simulation aléatoire et les arbres de recherches.

MCTS est un algorithme de recherche de type best-first c'est-à-dire que l'on cherche à identifier le nœud le plus prometteur, via une fonction d'évaluation, pour le parcourir en priorité.

Principe

Dans son application au poker, la méthode consiste à aléatoirement déterminer les valeurs des nœuds de décisions (enchères) et des nœuds de chance (donnes) de l'arbre de jeu jusqu'à atteindre une feuille ou le paiement est déterminé. En répétant l'opération un grand nombre de fois, on peut converger vers une bonne évaluation de l'espérance dans les nœuds intermédiaires sans avoir à parcourir l'entièreté de l'arbre.

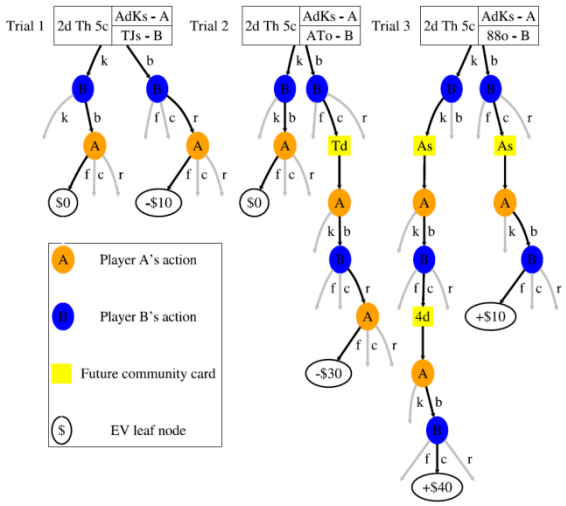

Figure 9 : Exemple d'échantillons dans un MCTS[20]

Processus

Un arbre de recherche se compose donc de :

- Feuilles (paiements)
- Nœuds de décision (enchères)
- Nœuds de chance (donnes)
- Nœuds de l'adversaire (enchères)

Il est construit nœud par nœud selon les résultats des évènements simulés du jeu. Chaque nœud doit contenir deux informations importantes :

- Une valeur estimée (espérance estimée) basée sur les résultats de la simulation
- Le nombre de fois ou le nœud a été visité (un compteur)

Le processus se divise en 4 étapes, il est répété jusqu'à la fin du temps de calcul défini :

1. Sélection

On démarre depuis le nœud racine R, récursivement on sélectionne un fils optimal (fonction de la stratégie de sélection) jusqu'à ce qu'une feuille L soit atteinte (cette feuille n'est pas nécessairement une feuille de l'arbre de jeu).

2. Expansion

Une ou plusieurs feuilles sont ajoutées à l'arbre comme fils de la feuille atteinte à l'étape précédente (si celle-ci n'était pas une feuille de l'arbre de jeu).

3. Simulation

Une simulation de jeu est effectuée à partir de la feuille ajoutée jusqu'à la fin. La valeur obtenue (de la feuille de l'arbre de jeu) est enregistrée. Une évaluation heuristique n'est pas nécessaire puisque chaque jeu est entièrement simulé.

4. Rétro propagation

L'estimation de l'espérance (expected value) ainsi que le compteur sont mis à jour dans tous les nœuds du chemin exploré (fonction de la stratégie de rétro propagation).

Après un nombre d'itérations de ce processus (dépendant du temps de calcul défini), une stratégie de sélection d'action est responsable de choisir la bonne action en fonction de l'espérance et de la valeur du compteur de chaque fils de la racine.

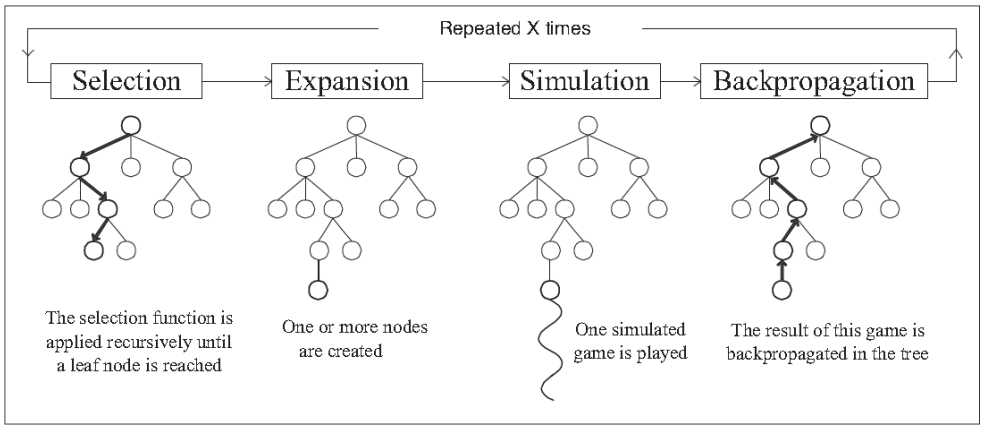

Figure 10 : Étapes du processus d'un MCTS (tiré de[4])

Stratégie de sélection

Les performances de cette technique sont très sensiblement liées à la stratégie de sélection utilisée. Il existe différentes stratégies comme :

- La sélection aléatoire qui maximise le nombre de visites,
- la recherche du meilleur compromis exploration/exploitation (dérivé du problème dit du bandit manchot),
- maximiser la probabilité de choisir la meilleure alternative après un nombre fini d'itérations (Algorithmes « selecting-the-best »).

4.4.3 Approximations de stratégies optimales

Les principes de la théorie des jeux dont nous avons parlé précédemment, nous offrent des outils pour la résolution des jeux.

Néanmoins la recherche de solutions optimales s'avère très rapidement ardue à mesure que la complexité augmente.

1. Optimisation linéaire

Les premières démarches dans ce sens étaient réalisées en plusieurs phases :

1. Diminuer la complexité par l'utilisation de plusieurs modèles (généralement un pré flop et un post flop) dans lesquels on a recours à différentes abstractions avec et sans pertes d'informations.
2. Utiliser la représentation en « sequence form[11] » afin d'assurer la conversion en un problème d'optimisation.
3. Appliquer un algorithme de résolution des problèmes d'optimisation linéaire comme le simplexe ou les points intérieurs.
4. Un équilibre ainsi obtenu pour le jeu abstrait est utilisé dans le jeu original ou il représente un ε-Nash.

Le problème est que cette approche nécessite une mémoire linéairement dépendante de la taille de l'arbre de jeu, ce qui ne permet pas de traiter les 4 phases (pré flop, flop, turn, river) en un modèle. Or, le fait de scinder le jeu original annule la garantie de trouver un équilibre de Nash.

2. Algorithmes itératifs

Fictitious play

La méthode « Fictitious play » se base sur l'idée très simple que si deux joueurs s'affrontent un grand nombre de fois, leur stratégie va s'adapter et s'améliorer au fur et à mesure de leurs confrontations.

L'algorithme peut être décomposé en 3 phases :

1. Chaque agent se voit attribuer de manière arbitraire une stratégie (un triplet [f, c, r] de probabilités pour chaque état du jeu). Les agents connaissent la stratégie de leur adversaire.

2. Une phase d'entrainement composée de situations choisies aléatoirement est accomplie. Les agents peuvent, en connaissance de la stratégie adverse, choisir la meilleure action dans une situation donnée.
3. Les agents mettent à jours leur stratégie de base avec les nouvelles informations.

Le nombre d'itérations augmentant les stratégies doivent converger vers un équilibre de Nash.

Range of Skill

L'algorithme de « Range of Skill[7] » consiste à créer une séquence d'agents dans laquelle chaque agent est capable de battre l'agent qui le précède dans la séquence d'au moins ε. Quand un nouvel agent ne parvient plus à atteindre cet objectif on a atteint un ε-Nash.

Pour améliorer la stratégie, cet algorithme fait appel à un autre algorithme appelé « generalised best response » qui calcule la meilleure réponse à un ensemble restreint de stratégies.

Excessive gap technique

L'EGP est un algorithme « anytime » (qui donne une solution valide à tout moment du processus) et qui ne requiert que O(1/ε) itérations pour approcher un ε-Nash. Il utilise la représentation « sequence form » et utilise directement la représentation en point-selle d'un équilibre de Nash dont l'équation est :

$$\max_{x} \min_{y} x^T A y = \min_{y} \max_{x} x^T A y$$

Avec :

- A la matrice de paiement du joueur 1
- x la stratégie du joueur 1
- y une stratégie du joueur 2

Cette technique a été la première à résoudre un modèle abstrait portant sur l'ensemble du jeu.

Counterfactual regret minimisation

L'algorithme CFR[23] est le nec plus ultra en matière d'algorithme pour la conception d'agents ε-Nash. C'est la technique utilisée par les vainqueurs de la catégorie « Bankroll Instant Run-Off » depuis plusieurs années.

L'avantage de cet algorithme est qu'il ne nécessite pas de stocker l'arbre de jeu en mémoire mais seulement de le traverser. Il ne requiert donc pas une mémoire proportionnelle au nombre d'états possibles du jeu mais seulement au nombre d'ensembles d'informations du jeu. Les ensembles d'informations qui représentent les informations visibles par un joueur sont stockés dans des arbres séparés (un par joueur).

Le CFR est un algorithme itératif qui minimise le regret c'est-à-dire la différence entre l'utilité de l'action choisie et l'utilité de la meilleure action. Si le maximum d'utilité est offert par l'action a^*, le regret d'un joueur choisissant l'action a vaut :

$$u(a^*) - u(a)$$

Le terme « Counterfactual » signifie que le calcul du regret est pondéré par la probabilité pour le joueur d'atteindre l'ensemble d'informations pour lequel le regret est calculé.

L'algorithme commence par assigner arbitrairement des stratégies aux joueurs. Ensuite, des jeux en information parfaite sont répétés entre les joueurs. Leurs stratégies respectives sont modifiées à la fin de chaque jeu afin de minimiser le regret. À mesure que le nombre de jeux augmente, les stratégies évoluent vers un équilibre de Nash.

Cet algorithme peut être combiné avec la méthode Monte Carlo[15].

4.4.4 Contre-stratégies d'exploitation

Les agents qui tentent d'exploiter les faiblesses de leur adversaire, s'éloigne par définition d'un équilibre de Nash et sont donc également exploitable.

Ce type d'agents construit en général un modèle de l'adversaire afin de l'évaluer et de l'exploiter.

1. ***Imperfect information game tree search***

Un algorithme comme minimax (ou plutôt expectimax son équivalent adapté aux jeux avec intervention du hasard) peut, grâce à l'apport d'un modèle de l'adversaire, combler les informations manquantes afin d'estimer l'espérance.

Deux algorithmes ont, sur cette idée, été développé par l'université d'Alberta.

Miximax

L'algorithme choisit par défaut l'action qui maximise l'espérance.

Miximix

L'algorithme choisit un compromis entre exploration et exploitation ce qui réduit également sa prédictibilité.

4.4.5 Contre-stratégies optimales

1. ***Frequentist best response***

Cette technique calcule une meilleure réponse pour un adversaire donné. Ce calcule est réalisé offline, c'est-à-dire qu'elle n'est pas réalisée pendant le cours du jeu mais par exemple sur des fichiers .log. Cette technique est statique et ne convient que pour un adversaire bien déterminé. Elle expose fortement l'agent à une exploitation.

Des techniques basées sur le même principe comme « Restricted Nash Response » ou « Data biased response » visent à diminuer l'exploitabilité de l'agent alors que « Teams of agents » vise à pouvoir exploiter différent styles d'adversaires.

4.4.6 Raisonnement à partir de cas

Le raisonnement à partir de cas (RàPC) consiste à retrouver dans une base de connaissances des cas analogues au problème en question ce qui ressemble assez fort au comportement humain. La base de connaissance est maintenue avec les couples problème/solution préalablement rencontrés.

L'application du RàPC au poker consiste à enregistrer des cas décrivant des manches jouées avec leur solution et le paiement lié. Les solutions de ces cas sont ensuite réutilisées pour aider à de futures décisions. La méthode des k plus proches voisins est utilisée pour retrouver les cas les plus proches.

4.4.7 Apprentissage supervisé

1. ***Réseau de neurones***

Parmi les différentes techniques d'apprentissage supervisé, les réseaux de neurones est sans doute celle qui le plus souvent été utilisé dans le domaine du poker, notamment afin de modéliser l'adversaire[6].

D'excellentes performances ont été réalisées par des agents utilisant cette technique qui est souvent préférées aux autres quand la complexité est très importante comme en full ring (10 joueurs).

4.4.8 Réseau bayésien

Un réseau bayésien est un modèle probabiliste sous la forme d'un graphe orienté acyclique dont les nœuds représentent des variables aléatoires. Chaque nœud dans le réseau est associé à une table de probabilités conditionnelles qui permet le calculer la valeur des probabilités en se basant sur la valeur des nœuds parent. Un diagramme d'influence peut être associé au réseau bayésien afin de modélisés et résoudre l'utilité maximale attendue.

5 LA COMPÉTITION

Afin de dynamiser les recherches sur l'IA appliquée au poker, l'association Américaine d'Intelligence Artificielle (AAAI) à décider d'organiser depuis 2006 une compétition de « Computer Poker » se déroulant dans le cadre de leur conférence annuelle.

Cette année la conférence se déroule à Bellevue (Seattle) Washington, USA, du 14 au 18 juillet, 2013

La compétition est organisée avec la variante Texas-Holdem limit et no-limit du Poker. Le règlement prévoit deux configurations ; Heads-up (deux joueurs) et 3 joueurs. Au total il existe 3 catégories :

- Heads-Up limit
- Heads-Up no-limit
- 3 joueurs limit

Nous allons maintenant passer en revue les informations les plus importantes concernant la compétition. Un document exhaustif (ACPC2013_1-1.pdf) se trouve sur le cd-rom accompagnant ce mémoire. Une partie des informations sont également disponibles sur le site de la compétition.

L'objectif étant de participer à la catégorie Heads-Up limit nous nous limiterons aux renseignements la concernant.

5.1 Règlement

5.1.1 Pratique

Les programmes doivent être remis le 2 juin au plus tard.

5.1.2 Jeu

Variante :	Texas Hold'em
Forme :	Fixed limit
Type :	Heads-up (2 joueurs)
Mode :	Cash game
Format :	Série de doubles matches
Matches par série :	40 doubles matches
Manches par match :	3000
Tapis :	Infini
Taille des mises :	10/20
Taille des blinds :	5/10
Structure des blinds :	Inverse
Actions illégales :	Interprétées comme un *CALL*
Vainqueur :	Bankroll Instant Run-Off & Total Bankroll

Tableau 5 : Tableau récapitulatif des règles du jeu

1. Structure

Blinds

La *sb* vaut 5 et la *bb* 10.

Il existe une particularité dans les parties de Heads-up, à savoir que la *sb*, qui est le dealer, parle en premier pré flop et en dernier lors des autres phases.

Mises

Dans les phases pré flop et flop la mise est de 10 (*SB*), elle est de 20 (*BB*) pour la turn et la river.

Le nombre de mises (*BET* et *RAISE*) est limité à 3 durant la phase pré flop et à 4 dans les autres phases.

Cash game

La taille des mises et des blinds est fixe (au contraire des tournois). Le tapis des joueurs est supérieur au montant maximum que l'on peut miser sur l'ensemble d'une manche (jamais nul) et il est réinitialisé au début de chaque manche.

2. Détermination du vainqueur

Chaque catégorie possède deux classements et donc deux vainqueurs.

Le premier classement (Total Bankroll) attribue la victoire à l'agent qui aura pris le plus de jetons à l'ensemble de ces adversaires. Ce classement vise à récompenser les agents qui maximisent l'exploitation de leurs adversaires.

Le second classement (Bankroll Instant Run-Off) attribue la victoire à l'agent qui aura survécu à un mécanisme éliminant à chaque cycle le joueur ayant gagné le moins de jetons face à l'ensemble de ces adversaires. Ce classement vise à récompenser les agents capables de battre les meilleures adversaires.

Un échantillon de mains destiné à réduire la variance est utilisé pour départager les robots. Chaque robot a 7 secondes pour donner sa décision.

Avec deux vainqueurs par catégorie, il existe donc six divisions et chaque équipe peut inscrire un programme par division.

3. Limite de perte

Afin d'éviter qu'un agent de niveau très faible ne soit trop déterminant dans les résultats, la perte maximale est limitée à 750 milli-big blinds par manche. Cette limite représente la perte qu'un agent subirait en choisissant l'action *FOLD* tout le temps.

4. Chumps

Le niveau de la compétition augmentant chaque année, il devient de plus en plus difficile d'exploiter les faiblesses des adversaires. Les agents basés sur une modélisation de l'adversaire et l'exploitation de ses faiblesses représen-

tent donc un intérêt moindre. Afin d'endiguer ce phénomène dans la catégorie Total Bankroll, des agents intentionnellement faibles appelés « chumps » sont introduit dans la compétition.

5. Double matche

Afin de minimiser l'effet de la variance chaque match d'une série est rejoué en inversant les cartes privatives des joueurs et ce pour toutes les manches.

Un match compte 3000 manches et il y a 40 doubles matches par série.

Une série (240.000 manches) permet, en général, de départager avec certitude deux agents.

5.1.3 Détails techniques

1. Hardware

Les agents participant à la compétition tournent sur des instances Amazon EC2 avec un espace disque de 100 GB.

Ci-dessous les caractéristiques de la meilleure machine virtuelle disponible.

Type :	M1 Large Instance
Mémoire :	7.5 GiB
Unité de calcul :	2 virtual core with 4 EC2 Compute Unit
Stockage :	100 GB
Plateforme :	64-bit
Système d'exploitation :	Linux
	http://aws.amazon.com/ec2/instance-types/

Tableau 6 : Configuration matérielle

2. Software

Afin de permettre aux concurrents de tester localement leur programme, le code du serveur de jeu ainsi que le protocole de communication est disponible sur le site de la compétition.

Les participants ont accès à leur machines (via SSH), au minimum une semaine avant la soumission finale.

Tous les fichiers nécessaires à la bonne marche du programme doivent être rassemblés dans un dossier sur la machine virtuelle de test. Ce dossier sera ensuite déplacé par les organisateurs vers la machine définitive.

Pour vérifier s'il tourne correctement et qu'il atteint le minimum de performance requis, le programme doit préalablement être validé pour pouvoir concourir. Cette étape est réalisée via un script disponible sur la machine de test.

3. Déroulement de la compétition

La compétition est automatisée.

Les programmes ne sont pas autorisés à se connecter à Internet.

A la moitié d'un double match, les machines sont remises à zéro et le programme réinstallé.

Le programme a au maximum 10 minutes pour démarrer.

Le serveur de jeu se connecte aux deux clients ou sont installés les programmes des agents.

Le temps total maximum qu'un agent a à sa disposition pour un match est de 7 secondes par manche. Donc pour un match de 3000 manches, l'agent dispose au total de 21.000 secondes. Le temps maximum pour une action est de 600 secondes.

Un dépassement du temps maximum entraine la disqualification de la compétition.

Génération des cartes

Les graines aléatoires (random seeds) qui seront utilisées pour l'initialisation des générateurs de nombres pseudo-aléatoires sont fournies par les participants.

5.1.4 Benchmark

Les participants de la compétition ont accès, pendant les douze mois qui suivent celle-ci, au benchmark server. Ce serveur donne l'opportunité de jouer des matches contre tous les agents engagés dans la catégorie du participant.

Ceci représente une excellente opportunité de test en conditions réelles.

6 LE CONCEPT

6.1 Objectif

L'objectif de la partie « pratique » de ce mémoire est double : d'une part, concevoir un bot possédant le niveau minimum requis pour participer à l'ACPC 2013. D'autre part, utiliser une technique innovante pour le domaine et ainsi participer quelque peu aux recherches.

6.1.1 Problème

Pour atteindre les performances minimum requises et pourquoi pas tenter d'être compétitif face aux autres agents, il faut être capable de s'approcher au mieux d'un équilibre de Nash (ε faible). Hors les solutions les plus performantes, comme la « Counterfactual Regret Minimization » peuvent être complexe à mettre en œuvre et nécessite des ressources importantes. La partie simulation de l'implémentation de Slumbot[9] (vainqueur de l'ACPC 2012 Heads-Up limit) représente 2880 heures soit 120 jours de calcul sur un cluster de 9 machines grand publiques.

L'utilisation d'une telle solution n'est donc pas envisageable, qui plus est, elle n'aurait pas de caractère innovant.

6.1.2 Solution

Un élément prépondérant dans l'orientation de mon choix est la disponibilité, sur le site de l'ACPC, des fichiers .log (voir annexe 1) de la compétition depuis 2009. Cela représente plusieurs centaines de millions de mains rien que pour la catégorie Heads-Up limit et donc une belle opportunité d'utiliser une méthode d'apprentissage supervisé.

Les réseaux de neurones ayant déjà été utilisés à plusieurs reprises, j'ai cherché une autre méthode pour finalement m'intéresser aux modèles probabilistes de Markov.

6.2 Modèles de Markov

Un modèle de Markov est un automate probabiliste à états finis qui se base sur la propriété de Markov pour laquelle les états futurs, ne dépendent que de l'état présent. Cela implique que toute information nécessaire à la prédiction parfaite du système doit être contenue dans l'état actuel du modèle.

6.2.1 Les chaînes de Markov

Dans une chaîne de Markov, on fait l'hypothèse qu'il y a plusieurs évolutions possibles à partir de la situation présente, chacune d'elles ayant une certaine probabilité de se réaliser.

On peut matérialiser cela par un graphe d'état doté d'une fonction de transition probabiliste.

$S : \textit{Ensemble fini d'états}$

$A : \textit{Fonction de transition} : S \times S \Rightarrow [0; 1]$

6.2.2 Processus de décision markovien (MDP)

Un modèle de décision Markovien ou MDP (Markov decision process) peut être vu comme une chaîne de Markov à laquelle on ajoute une composante décisionnelle ainsi que la notion de récompense.

$S : \textit{Ensemble fini d'états}$

$I : \textit{Ensemble fini d'actions}$

$A^* : \textit{Fonction de transition} : S \times I \times S \Rightarrow [0; 1]$

$R : \textit{Fonction de récompense} : S \times I \Rightarrow R$

On peut définir l'objectif du système comme le choix d'une action, dans l'état actuel, qui maximise la récompense.

6.2.3 Modèles de Markov cachés (HMM)

Un modèle de Markov caché ou HMM (Hidden Markov model) est une autre évolution possible des chaînes de Markov. Ce nouveau modèle est basé sur deux processus stochastiques interdépendant. L'état du système n'est plus directement observable, caché par un processus d'observation.

$S : Ensemble\ fini\ d'états$

$A : Fonction\ de\ transition : S \times S \Rightarrow [0;1]$

$B : Fonction\ d'observation : S \times O \Rightarrow [0;1]$

6.2.4 Processus de décision markovien partiellement observable (POMDP)

Un Processus de décision markovien partiellement observable ou POMDP (Partially Observable Markov Decision Process) sont issus de la fusion des MDP et des HMM. Ils cumulent la notion d'observation et celle d'action engendrant une double incertitude :

- L'effet des actions que l'on entreprend est incertain (actions probabilistes),
- On ne dispose que d'observation pour connaitre l'état dans lequel on se trouve.

6.2.5 Modèle de Maximum d'entropie (MaxEnt)

Un modèle de Maximum d'entropie ou MaxEnt (maximum entropy model) est un classifieur probabiliste linéaire et discriminant.

« Le principe d'entropie maximale vise à définir une contrainte pour chaque information observée et choisir la distribution qui maximise l'entropie (l'incertitude) tout en restant consistante vis-à-vis de l'ensemble de ces contraintes » [10].

Un MaxEnt est élaboré en trois étapes :

- Créer un corpus d'entrainement relatif à un phénomène aléatoire

- Identifier les traits qui décrivent le phénomène aléatoire
- Choisir le modèle qui maximise l'entropie mais respecte les contraintes

6.2.6 Modèles de Markov de maximum entropie (MEMM)

Un modèle de Markov de maximum entropie ou MEMM (Maximum-entropy Markov model) combine les caractéristiques des HMM et des MaxEnt. Ce modèle[17] permet de représenter des observations comme des caractéristiques superposées et définis la probabilité conditionnelle de séquences d'états étant donné les séquences d'observations.

6.2.7 Champs aléatoires conditionnels (CRF)

Les champs aléatoires conditionnels ou CRF (Conditional Random Fields) sont une méthode d'extraction d'informations présentée par Lafferty et al.[14] en 2001 qui souhaitait lever certaines limitations liées aux HMM et aux MEMM. Les CRF sont notamment utilisés dans :

- le traitement automatique du langage naturel,
- la vision par ordinateur,
- la bio-informatique.

« Un CRF est un cadre pour construire des modèles probabilistes afin de segmenter et labéliser des séquences de données. »

Plus concrètement, le CRF modélise une distribution de probabilités conditionnelles sur un ensemble de variables cibles étant donné un ensemble de variables observées.

PARTIE 2 : PRATIQUE

7 ANALYSE

L'objectif de la partie pratique est de créer un agent afin de participer à l'ACPC 2013 dans la catégorie Heads-up limit. Nous avons décidé d'utiliser les fichiers .log des années précédentes comme base pour un apprentissage supervisé.

1. Extraction de l'information

L'information contenue dans les fichiers .log doit être traitée pour être utilisable. Le traitement de l'information doit également assurer la validité de celle-ci (détection d'erreur visible dans le fichier). Il est donc nécessaire d'implémenter un « parser » .

Afin de s'inspirer des meilleurs agents, nous extrayons l'information des 3 premiers de la compétition 2012 en « Bankroll Instant Run-off » à savoir Slumbot (Eric Jackson, USA), Hyperborean (University of Alberta, Canada) et Zbot (Ilkka Rajala, Finland).

2. Traitement de l'information et modélisation

Une fois l'information extraite, il faut réussir à faire ressortir les stratégies sous-jacentes des agents et les intégrer dans un ou plusieurs modèles. Nous nous servirons pour cela d'un logiciel d'apprentissage supervisé ; Wapiti.

3. Framework

Outre l'aspect « intelligence » l'agent doit également être capable de communiquer avec le serveur de jeu en respectant le temps imparti.

8 IMPLÉMENTATION

8.1 Parser

L'implémentation du « parser » est réalisée en Java.

Par agent, il y 3000 fichiers contenant chacun 3000 manches, soit un total de 9.000.000 de manches à traiter.

Le parser reçoit en paramètre le chemin d'un répertoire et le nom d'un agent.

Les fichiers à traiter sont identifiés grâce à leur nom qui est du type :

```
2pl.nomAgent1.nomAgent2.0.0.log
```

Une fois le fichier identifié, le premier objectif est de s'assurer que la ligne du fichier .log concerne bien un résultat et qu'elle ne contient pas d'erreur visible. Ceci est réalisé grâce à l'utilisation d'une expression régulière.

Une fois la ligne validée, on procède à l'extraction des informations.

Prenons comme exemple la ligne ci-dessous.

```
STATE:0:rrc/rc/crc/crf:9cKc|7s8d/6dJs5c/7d/Ts:-60|60:slumbot|entropy
```

La ligne se divise en 4 parties :

1. Les actions [pré flop/flop/turn/river]
2. Les cartes [bb|sb/flop/turn/river]
3. Le paiement [bb|sb]
4. La position [bb|sb]

Le joueur 1 est en big blind et le joueur 2 (dealer) est en small blind.

Nous devons extraire les informations visibles par l'agent « slumbot » ainsi que toutes les actions réalisées dans chaque phase. Nous savons que slumbot est en *bb* et qu'il parlera en second pré flop et en premier dans les autres phases.

En sortie du parser nous obtenons le résultat ci-dessous.

```
pflop  blind  r   -   -   -   K   9   -   -   -   -   -   a   -   r
flop   blind  rrc -   -   -   K   9   J   6   5   -   -   ega King High r
turn   blind  rrc rc  -   -   K   9   J   7   6   5   -   egga  King High c
turn   blind  rrc rc  cr  -   K   9   J   7   6   5   -   egga  King High c
river  blind  rrc rc  crc -   K   9   J   T   7   6   5   -   King High c
river  blind  rrc rc  crc cr  K   9   J   T   7   6   5   -   King High f
```

Figure 11 : Représentation d'une séquence.

Chaque ligne correspond donc aux informations disponibles et à la décision prise par l'agent (dernière colonne à droite).

Afin d'améliorer la qualité du modèle généré par la suite, le parser réalise également une abstraction sur l'enseigne (a, ega, egga) et détermine la valeur de la combinaison (ici hauteur roi).

La stratégie au niveau du jeu pré flop est très facilement identifiable, nous pouvons dès lors, via le parser, la stocker directement dans plusieurs matrices 13x13 appelées « starting hand chart » (un exemple est visible en annexe 2). L'agent pourra immédiatement l'utiliser pour le jeu pré flop.

8.2 Modélisation

8.2.1 Logiciel

Le programme utilisé pour générer les modèles de jeu dans le cadre de ce travail est Wapiti[16], il s'agit d'un logiciel sous licence BSD qui propose un ensemble d'outils pour la segmentation et l'étiquetage de séquences. Il est basé sur les modèles de maximum entropie, les modèles de Markov de maximum entropie (MEMM) et les CRF. Il est développé par LIMSI, un laboratoire du CNRS.

1. Algorithmes d'entrainement

Plusieurs algorithmes d'entrainement sont disponibles :

- L-BFGS et OWL-QN (Méthode de Quasi-Newton),
- SGD-L1 (Méthode du gradient),
- BCD (Block Coordinate Descent),
- R-PROP (Méthode de premier ordre).

2. Utilisation

Plusieurs modes sont disponibles mais seul le mode entrainement nous intéresse dans un premier temps (le mode label sera utilisé dans l'agent ou il déterminera les actions à effectuer par celui-ci). Dans ce mode le logiciel attend en entrée un corpus d'entrainement (train dataset), un corpus de validation (devel dataset) et un pattern. Il produit en sortie un modèle optimisé dont le taux d'erreurs est calculé sur le corpus de validation.

Commande

Ci-dessous un exemple de ligne de commande :

```
wapiti train [options] [input data] [model file]

wapiti train –c –t 8  –a rprop –p pattern.txt slumbotTrain.txt
–d slumbotDevel.txt slumbot.model
```

Avec :

- train : mode entrainement
- –c : mode compacte (la taille du modèle en sortie est optimisée)
- –t 8 : nombre de thread à utiliser (influence sensiblement la performance)
- –a rprop : nom de l'algorithme à utiliser
- –p pattern.txt : nom du fichier contenant le pattern
- slumbotTrain.txt : nom du fichier contenant le corpus d'entrainement
- –d slumbotDevel.txt : nom du fichier contenant le corpus de validation
- slumbot.model : nom du modèle à créer

Pattern

Le fichier de pattern indique au logiciel comment agencer l'information contenue dans le corpus.

```
# Round P or F or T
u:Round=%x[0,0]

# Actions P & F & T
*:AAc=%x[0,2]/%x[0,3]/%x[0,4]/%x[0,5]

#Hole cards
u:HC=%x[0,6]/%x[0,7]

#board
*:Board=%x[0,8]/%x[0,9]/%x[0,10]/%x[0,11]/%x[0,12]

# hole cards + board
*:Board=%x[0,6]/%x[0,7]/%x[0,8]/%x[0,9]/%x[0,10]/%x[0,11]/%x[0,12]

# Suit pattern
u:SP=%x[0,13]

# HC + Suit pattern
*:HSP=%x[0,6]/%x[0,7]/%x[0,13]

# hand name
*:HN=%x[0,14]
```

Figure 12 : Fichier pattern de Wapiti

Toute ligne vide ou commençant par "#" est ignorée par le logiciel.

Une ligne doit commencer par un des caractères suivant qui spécifie le type de trait :

- 'u' : unigram
- 'b' : bigram
- '*' : unigram et bigram

Un trait est spécifié par %x[off,col] où "col" est le numéro de la colonne et "off" est l'offset éventuel.

Pour faciliter la lecture, les traits seront combinés avec le caractère "/".

Informations

Durant la modélisation les informations sont communiquées à l'utilisateur via le terminal.

```
* Load patterns
* Load training data
   1000 sequences loaded
   2000 sequences loaded
   3000 sequences loaded
...
4769000 sequences loaded
4770000 sequences loaded
4771000 sequences loaded
* Load development data
   1000 sequences loaded
   2000 sequences loaded
   3000 sequences loaded
...
 528000 sequences loaded
 529000 sequences loaded
 530000 sequences loaded
* Initialize the model
* Summary
    nb train:    4771728
    nb devel:    530725
    nb labels:   3
    nb blocks:   323160
    nb features: 3876993
* Train the model with rprop
  [   1] obj=5242279.02 act=0        err=45.41%/45.41% time=0.93s/0.93s
  [   2] obj=5242279.02 act=869516   err=31.74%/31.74% time=0.92s/1.85s
  [   3] obj=3807120.66 act=694435   err=38.53%/38.53% time=0.94s/2.78s
...
  [  47] obj=1675944.03 act=822459   err=12.63%/12.63% time=0.98s/44.88s
  [  48] obj=1675634.73 act=822535   err=12.64%/12.64% time=0.95s/45.83s
  [  49] obj=1675326.94 act=822649   err=12.64%/12.64% time=0.95s/46.78s
* Compacting the model
    - Scan the model
    - Compact it
       10310 observations removed
      123720 features removed
* Save the model
* Done
test_Model Finished
```

Figure 13 : Informations communiquées par Wapiti durant une modélisation

L'information la plus importante est le taux d'erreur du modèle (err=x%), c'est-à-dire le taux d'erreur du modèle évalué sur le corpus d'évaluation.

8.2.2 Modélisation

1. Corpus et pattern

En sortie du parser, nous avons un fichier contenant l'information mise en forme concernant un agent pour les phases flop, turn et river.

À partir de ce fichier on crée le corpus d'entrainement qui représente 90% du fichier de départ et le corpus de validation avec les 10% restant (la proportion est choisie par l'utilisateur).

La première étape a consisté à définir les informations du corpus et la manière de les combiner via le pattern. Elle a été réalisée via de nombreux tests visant à minimiser le taux d'erreur. Le pattern finalement adopté est représenté en figure 12.

L'idée de départ était d'utiliser des séquences via la fonction CRF de Wapiti. Une séquence dans notre cas représente un ensemble de lignes concernant une même manche comme représenté sur la figure 11. À l'usage, le fait de regrouper les décisions en séquences n'a rien apporté en termes de minimisation du taux d'erreurs mais s'avère par contre bien plus délicat au niveau de la consommation de ressources (CPU et surtout mémoire vive). Le corpus a donc été modifié de manière à avoir une séquence pour chaque décision et nous travaillons donc en mode maximum entropie.

La seconde étape a été de créer différents modèles à partir des trois agents choisis (slumbot, Hyperborean et Zbot) dans le but de les tester en configuration réelle c'est-à-dire sur le serveur de jeu.

Pour chaque agent nous souhaitons tester les types de modèles suivants :

- Un modèle comprenant les 3 phases (flop, turn et river) : agent_ALL.model
- Un modèle uniquement pour la phase flop : agent_FLOP.model
- Un modèle uniquement pour la phase turn : agent_TURN.model
- Un modèle uniquement pour la phase river : agent_RIVER.model

Afin de déterminer l'influence de l'augmentation du volume du corpus, des types de modèles multi-agents ont également un intérêt (slumbot + Hyperborean et slumbot + Hyperborean + Zbot).

Pour chaque type de modèle, un modèle est créé par algorithme (sauf pour l'algorithme BCD qui ne semble pas fonctionner sur des corpus de cette taille).

Ci-dessous le tableau récapitulatif des 45 modèles et de leur taux d'erreurs.

	l-bfgs	**Sgd-l1**	**rprop**
Slumbot All	24,70%	14,55%	14,66%
Slumbot Flop	19,23%	15,89%	15,89%
Slumbot Turn	15,00%	13,97%	14,09%
Slumbot River	13,70%	12,57%	12,63%
Hyperborean All	25,15%	16,75%	16,84%
Hyperborean Flop	21,05%	17,71%	17,71%
Hyperborean Turn	27,68%	16,07%	16,21%
Hyperborean River	16,57%	15,50%	15,63%
Zbot All	24,92%	17,33%	17,43%
Zbot Flop	20,80%	17,85%	17,84%
Zbot Turn	19,07%	18,28%	18,42%
Zbot River	16,19%	15,21%	15,34%
Slumborean Flop	20,05%	16,81%	16,78%
Slumborean Turn	16,35%	15,53%	16,65%
Zblumborean River	15,87%	15,01%	15,17%

Figure 14 : Tableau des modèles avec taux d'erreurs par algorithme

Erreurs

Il faut signaler que les taux d'erreurs du tableau ci-dessus sont surévalués. En effet, quand le logiciel effectue la validation il ne tient évidemment pas compte du fait que l'agent utilise une stratégie mixte. Si l'agent ne choisit pas l'action estimée comme la plus probable par le logiciel, une erreur est comptabilisée.

8.3 Framework

Afin de faciliter l'implémentation, la plateforme AI Poker Bot[12] a été utilisée.

Il s'agit d'une plateforme de développement d'agent de « computer bot » écrite en Java par deux étudiants de l'université de Brême (DE) en 2010. Elle supporte le Texas Hold'em limit et no-limit et sa structure permet, via l'ajout de modules, de facilement l'étendre à d'autres variantes de Poker mais surtout à d'autres techniques d'intelligence artificielles.

Elle est à l'origine conçue pour interagir avec le server ACPC 2010 et la première tâche a donc été de modifier le module de communication (classe Talk) pour l'adapter au server ACPC 2011 (protocole 2.0.0).

Ensuite l'implémentation c'est concentrée sur la partie « décisions » de l'agent via un « extends » de la classe abstraite Bot.

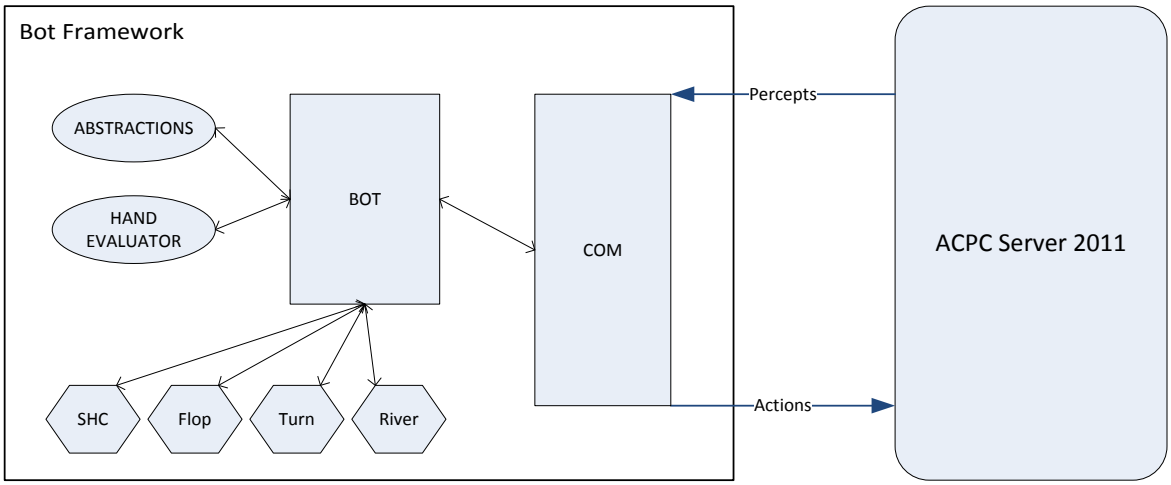

Figure 15 : Framework de l'agent

Comme montré dans la figure 15, l'agent communique avec le serveur. Il utilise, quand c'est nécessaire, les modules « abstractions » et « hand evaluator » pour formater l'information avec laquelle il va interroger les modèles (SHC, Flop, Turn et River). Le modèle SHC (starting hand chart) est interrogé durant la phase pré flop. Les modèles Flop, Turn et River sont les modèles générés par Wapiti. Comme on l'a vu précédemment, il peut également s'agir d'un modèle unique pour les 3 phases post flop.

9 TESTS

9.1 Cycles de sélection

Un très grand nombre de modèles ont été générés pour les différentes phases.

9.1.1 Modèles

1. ***Pré flop***

Sous forme de SHC (starting hand chart) nous avons les stratégies mixtes d'Hyperborean, de Slumbot et de Zbot. Auxquelles on ajoute celle calculée par Selby (voir annexe 2) dans le cadre d'un jeu équivalent au Texas Hold'em limit Heads-up à la différence qu'aucune action n'est possible post flop.

2. ***Post flop***

Post flop nous avons les 45 modèles construits avec Wapiti.

9.1.2 Mode de sélection

Afin de choisir les agents qui seront présentés à l'ACPC, les différentes configurations possibles seront départagées dans des doubles matches de 10.000 manches, soit au total 20.000 manches par confrontation.

L'agent qui sera choisi pour la catégorie « Bankroll Instant Run Off » sera celui capable de batte le plus grand nombre d'autres agents.

Pour la catégorie « Total Bankroll », l'accent sera mis sur les capacités d'exploitation (Un agent de niveau faible servira d'étalon).

9.1.3 Sélection 1 : Type de modèle post flop

La première sélection a opposé les agents (issus d'un même algorithme et d'un même agent original) possédant un modèle post flop unique et ceux possédant un modèle par phase (flop, turn et river). Les agents utilisent leur propre SHC.

Les résultats sont flagrants et quasi tous les agents à modèle post flop unique sont éliminés.

9.1.4 Sélection 2 : Type d'algorithme

La deuxième sélection a opposé les agents issus d'un même agent original mais d'algorithme différents.

Ici aussi les résultats sont sans appel et les agents générés par l'algorithme RPROP démontrent une nette supériorité sur les autres. Seul un agent issu de l'algorithme LBFGS parvient à tirer son épingle du jeu par ses qualités d'exploitation.

9.1.5 Sélection 3 : Agents original et SHC

Les agents rescapés des deux premières sélections (un agent par agent original plus le modèle multi-agents) sont tous issus de l'algorithme RPROP sauf un LBFGS et possède un modèle Wapiti par phase post flop.

Chaque agent encore en lice va être décliné en 4 agents, et chacun recevra une SHC différentes (sa propre SHC, celles des autres agents et celle de Selby).

Les résultats montrent que pour tous les agents, la SHC de Selby améliore les performances. L'agent qui remporte la sélection en battant la totalité des autres agents est le modèle post flop multi-agents et pré flop SHC de Selby.

Le modèle qui démontre les meilleures qualités d'exploitation est le modèle post flop slumbot et pré flop SHC de Selby.

9.2 Validation

Une fois les modèles sélectionnés, la dernière démarche à accomplir était de copier les codes sur la machine virtuelle de test de l'ACPC et de passer la validation pour les deux agents.

Aucun problème n'a été rencontré et les agents ont parfaitement passé cette dernière étape.

9.3 Analyse et enseignements

Après analyse des résultats des sélections, il apparait une supériorité des agents :

- Utilisant un modèle pré flop basé sur la SHC de Selby,
- utilisant un modèle par phase post flop (flop, turn, river),
- utilisant un modèle basé sur un corpus de très grande taille (corpus multi-agents),
- utilisant un modèle généré par l'algorithme d'entrainement RPROP.

Le taux d'erreurs calculé par Wapiti n'agit pas de manière sensible sur les résultats.

Il faut souligner que la génération d'un modèle avec Wapiti (en utilisant l'algorithme RPROP) est assez rapide et pourrait représenter une solution pour de la modélisation d'adversaire en cours de partie.

10 RÉSULTATS

10.1 Participants

Les participants 2013 de la catégorie « Heads-up Limit Texas Hold'em » sont au nombre de 12. Ci-dessous un aperçu de ceux-ci :

Agent	Concepteur	Affiliation	Pays	Technique(s)
Feste	F. Pays	Indépendant	France	Excessive gap technique
HITSZ_CS_13	X. Wang	HIT	Chine	Formula based
Hyperborean2pl.iro	M. Bowling	U. of Alberta	Canada	Chance Sampled CFR
Hyperborean2pl.tbr	M. Bowling	U. of Alberta	Canada	Data biased resp. + CFR
LIACC	LF. Teofilo	U. of Porto	Portugal	EV maximization
Little Rock	R. Byrnes	Indépendant	Australie	Monte Carlo CFR
Marv	M. Andersen	Indépendant	UK	Neural network
Neo Poker Bot	A. Lee	Neo Poker Laboratory	Espagne	
ProPokerTools	D. Hutchings	ProPokerTools	USA	CFR
Slugathorus	D. Berger	U. of New SouthWales	Australie	Monte Carlo CFR
Unamur	N. Verbeiren	U. of Namur	Belgique	Maximum entropy
Zbot	I. Rajala	Indépendant	Finland	CFR

Tableau 7 : Tableau des participants à l'ACPC 2013 (Heads-up limit)

10.2 Résultats

Les résultats sont donnés en milli big blind par manche et associés à un intervalle de confiance.

10.2.1 Bankroll Instant Run-off

	marv	feste_iro	hyperborean_iro	zbot	littlerock	neo_poker_lab	propokertools	HITSZ_CS_13	unamur_iro	liacc	slugathorus	AVG
marv		56,4 ± 9,4	-15,2 ± 5,7	-1,9 ± 6,2	9,2 ± 7,1	-10,7 ± 6,5	16,7 ± 7,1	231,3 ± 9,8	166,5 ± 8,2	523,0 ± 15,7	677,0 ± 12,1	*165,2*
feste_iro	-56,4 ± 9,4		-59,7 ± 9,7	-51,5 ± 10,3	-51,1 ± 8,5	-53,3 ± 8,9	-39,0 ± 8,3	156,1 ± 11,0	157,8 ± 7,9	643,8 ± 15,9	672,4 ± 18,4	*131,9*
hyperborean_iro	15,2 ± 5,7	59,7 ± 9,7		9,8 ± 4,3	21,6 ± 7,3	-11,2 ± 6,2	27,3 ± 8,2	157,0 ± 10,9	167,8 ± 9,2	390,1 ± 15,4	440,8 ± 15,3	127,8
zbot	1,9 ± 6,2	51,5 ± 10,3	-9,8 ± 4,3		13,4 ± 7,3	-6,5 ± 7,0	8,5 ± 7,8	169,0 ± 9,9	170,9 ± 7,1	385,3 ± 15,8	448,8 ± 14,9	123,3
littlerock	-9,2 ± 7,1	51,1 ± 8,5	-21,6 ± 7,3	-13,4 ± 7,3		-21,2 ± 8,1	5,5 ± 8,4	151,3 ± 8,3	160,0 ± 9,3	392,5 ± 19,2	469,3 ± 15,2	116,4
neo_poker_lab	10,7 ± 6,5	53,3 ± 8,9	11,2 ± 6,2	6,5 ± 7,0	21,2 ± 8,1		20,2 ± 6,9	150,5 ± 9,4	169,0 ± 10,0	313,6 ± 17,6	405,0 ± 18,6	116,1
propokertools	-16,7 ± 7,1	39,0 ± 8,3	-27,3 ± 8,2	-8,5 ± 7,8	-5,5 ± 8,4	-20,2 ± 6,9		159,5 ± 10,2	156,5 ± 9,6	402,2 ± 17,9	466,2 ± 16,9	114,5
HITSZ_CS_13	-231,3 ± 9,8	-156,1 ± 11,0	-157,0 ± 10,9	-169,0 ± 9,9	-151,3 ± 8,3	-150,5 ± 9,4	-159,5 ± 10,2		109,4 ± 10,8	609,3 ± 15,4	750,0 ± 0,0	29,4
unamur_iro	-166,5 ± 8,2	-157,8 ± 7,9	-167,8 ± 9,2	-170,9 ± 7,1	-160,0 ± 9,3	-169,0 ± 10,0	-156,5 ± 9,6	-109,4 ± 10,8		77,3 ± 16,4	324,6 ± 14,0	-85,6
liacc	-523 ± 15,7	-643,8 ± 15,9	-390,1 ± 15,4	-385,3 ± 15,8	-392,5 ± 19,2	-313,6 ± 17,6	-402,2 ± 17,9	-609,3 ± 15,4	-77,3 ± 16,4		750,0 ± 0,0	-298,7
slugathorus	-677 ± 12,1	-672,4 ± 18,4	-440,8 ± 15,3	-448,8 ± 14,9	-469,3 ± 15,2	-405,0 ± 18,6	-466,2 ± 16,9	-750,0 ± 0,0	-324,6 ± 14,0	-750,0 ± 0,0		-540,4

ROUND 1		ROUND 2		ROUND 3		ROUND 4		ROUND 5		ROUND 6		ROUND 7		ROUND 8		ROUND 9		ROUND 10	
marv	10,000	marv	9,000	marv	7,446	marv	6,534	neo_poker_lab	5,610	neo_poker_lab	4,789	neo_poker_lab	3,935	neo_poker_lab	2,972	neo_poker_lab	1,987	neo_poker_lab	0,999
feste_iro	8,886	hyperborean_iro	7,994	neo_poker_lab	6,793	hyperborean_iro	5,826	hyperborean_iro	5,389	hyperborean_iro	4,211	hyperborean_iro	3,065	hyperborean_iro	2,026	hyperborean_iro	1,003	hyperborean_iro	0,001
hyperborean_iro	8,089	zbot	6,905	hyperborean_iro	6,742	neo_poker_lab	5,634	zbot	3,567	zbot	2,742	zbot	1,926	zbot	0,913	zbot	0,010		
zbot	7,021	neo_poker_lab	6,088	zbot	5,019	zbot	4,006	marv	3,434	marv	2,258	marv	1,074	marv	0,089				
littlerock	5,263	littlerock	4,571	littlerock	3,844	littlerock	2,824	littlerock	1,922	littlerock	0,832	littlerock	0,000						
neo_poker_lab	5,214	propokertools	4,053	propokertools	3,156	propokertools	2,176	propokertools	1,078	propokertools	0,168								
propokertools	4,527	feste_iro	3,389	feste_iro	2,000	feste_iro	1,000	feste_iro	0,000										
HITSZ_CS_13	3,000	HITSZ_CS_13	2,000	HITSZ_CS_13	1,000	HITSZ_CS_13	0,000												
unamur_iro	2,000	unamur_iro	1,000	unamur_iro	0,000														
liacc	1,000	liacc	0,000																
slugathorus	0,000																		

1. *Podium final*

- Neo Poker Lab
- Hyperborean2pl.iro
- Zbot

2. *Résultat*

La version « Bankroll Instant Run-off » de UNamur termine 9ème sur 11, et est éliminé au 3ème tour. Il perd en moyenne 0,086 big blind par manche.

10.2.2 Total Bankroll

	marv	feste_tbr	hyperborean_tbr	zbot	littlerock	propokertools	neo_poker_lab	HITSZ_CS_13	chump12	unamur_tbr	liacc	chump1	slugathorus	AVG
marv		69,7 ± 8,8	36,8 ± 7,8	-1,9 ± 6,2	9,2 ± 7,1	16,7 ± 7,1	-10,7 ± 6,5	231,3 ± 9,8	218,1 ± 10,8	212,2 ± 8,9	523,0 ± 15,7	417,3 ± 10,2	677,5 ± 12,1	***199,9***
feste_tbr	-69,7 ± 8,8		-39,2 ± 9,6	-64,1 ± 10,3	-57,1 ± 9,2	-57,6 ± 10,5	-62,0 ± 9,4	140,8 ± 10,9	118,3 ± 11,1	198,4 ± 10,0	750,0 ± 0,0	539,5 ± 16,2	750,0 ± 0,0	***178,9***
hyperborean_tbr	-36,8 ± 7,8	39,2 ± 9,6		-25,7 ± 9,5	-10,7 ± 8,7	-8,0 ± 9,4	-31,8 ± 9,8	159,0 ± 9,4	138,6 ± 11,6	182,2 ± 11,9	527,0 ± 20,7	548,2 ± 16,5	617,4 ± 19,6	***174,9***
zbot	1,9 ± 6,2	64,1 ± 10,3	25,7 ± 9,5		13,4 ± 7,3	8,5 ± 7,8	-6,5 ± 7,0	169,0 ± 9,9	152,1 ± 10,1	187,2 ± 10,1	385,3 ± 15,8	365,5 ± 11,1	448,8 ± 14,9	***151,3***
littlerock	-9,2 ± 7,1	57,1 ± 9,2	10,7 ± 8,7	-13,4 ± 7,3		5,5 ± 8,4	-21,2 ± 8,1	151,3 ± 8,3	154,7 ± 9,3	169,3 ± 10,3	392,5 ± 19,2	366,7 ± 12,6	469,3 ± 15,2	***144,4***
propokertools	-16,7 ± 7,1	57,6 ± 10,5	8,0 ± 9,4	-8,5 ± 7,8	-5,5 ± 8,4		-20,2 ± 6,9	159,5 ± 10,2	151,9 ± 9,7	170,8 ± 9,3	402,2 ± 17,9	352,5 ± 12,2	466,2 ± 16,9	***143,2***
neo_poker_lab	10,7 ± 6,5	62,0 ± 9,4	31,8 ± 9,8	6,5 ± 7,0	21,2 ± 8,1	20,2 ± 6,9		150,5 ± 9,4	158,1 ± 14,1	183,6 ± 9,6	313,6 ± 17,6	352,3 ± 15,4	405,0 ± 18,6	***143,0***
HITSZ_CS_13	-231,3 ± 9,8	-140,8 ± 10,9	-159,0 ± 9,4	-169,0 ± 9,9	-151,3 ± 8,3	-159,5 ± 10,2	-150,5 ± 9,4		126,8 ± 20,5	213,4 ± 11,2	609,3 ± 15,4	542,9 ± 19,3	750,0 ± 0,0	***90,1***
chump12	-218,1 ± 10,8	-118,3 ± 11,1	-138,6 ± 11,6	-152,1 ± 10,1	-154,7 ± 9,3	-151,9 ± 9,7	-158,1 ± 14,1	-126,8 ± 20,5		93,3 ± 12,1	68,4 ± 16,8	677,0 ± 13,7	670,7 ± 15,2	***24,2***
unamur_tbr	-212,2 ± 8,9	-198,4 ± 10,0	-182,2 ± 11,9	-187,2 ± 10,1	-169,3 ± 10,3	-170,8 ± 9,3	-183,6 ± 9,6	-213,4 ± 11,2	-93,3 ± 12,1		132,3 ± 16,7	321,3 ± 12,6	437,0 ± 17,6	***-60,0***
liacc	-523,0 ± 15,7	-750,0 ± 0,0	-527,0 ± 20,7	-385,3 ± 15,8	-392,5 ± 19,2	-402,2 ± 17,9	-313,6 ± 17,6	-609,3 ± 15,4	-68,4 ± 16,8	-132,3 ± 16,7		602,5 ± 16,7	750,0 ± 0,0	***-229,3***
chump1	-417,3 ± 10,2	-539,5 ± 16,2	-548,2 ± 16,5	-365,5 ± 11,1	-366,7 ± 12,6	-352,5 ± 12,2	-352,3 ± 15,4	-542,9 ± 19,3	-677,0 ± 13,7	-321,3 ± 12,6	-602,5 ± 16,7		390,0 ± 19,3	***-391,3***
slugathorus	-677,5 ± 12,1	-750,0 ± 0,0	-617,4 ± 19,6	-448,8 ± 14,9	-469,3 ± 15,2	-466,2 ± 16,9	-405,0 ± 18,6	-750,0 ± 0,0	-670,7 ± 15,2	-437,0 ± 17,6	-750,0 ± 0,0	-390,0 ± 19,3		***-569,3***

1. *Podium final*

- Marv
- Feste
- Hyperborean2pl.tbr

2. Résultat

La version « Total Bankroll » de UNamur termine 10ème sur 13. Il perd en moyenne 0,06 big blind par manche.

CONCLUSION

Tout d'abord, si l'on reprend les objectifs de départ qui étaient d'explorer les applications de l'intelligence artificielle au poker et d'implémenter un agent capable de participer à l'ACPC, on peut dire que de ce point de vue le contrat est rempli.

Ensuite, pour ce qui est d'évaluer la technique utilisée, ce n'est pas chose aisée tant les résultats sont à nuancer. Il est vrai que les performances lors de la compétition ne sont pas exceptionnelles, mais elles ne sont finalement pas mauvaises non plus face aux meilleurs agents. Et ces performances sont-elles le reflet de la valeur des techniques ou plutôt de la manière dont je les ai exploitées ?

Enfin, d'un point de vue personnel cette expérience peut être qualifiée, et je pèse mes mots, d'exceptionnelle. En effet, jamais il y a un an je n'aurais cru pouvoir présenter un agent capable de jouer au poker dans une compétition rassemblant les meilleures programmes du domaine et ce sans même finir dernier. Mes connaissances en technique d'intelligence artificielle étaient proches de zéro et avec le recul l'idée que je me faisais de mon niveau en informatique était largement surestimée.

Au moment de terminer ce travail, je ressens donc une certaine satisfaction, mais surtout une soif d'en apprendre encore plus sur le sujet et je dois bien l'admettre un léger besoin de revanche.

PERSPECTIVES

Si je devais répondre à la question, quels sont les choses que j'améliorerais en priorité si j'avais plus de temps, je répondrais en 3 points :

- Je présenterais en plus une analyse du poker d'un point de vue logique,
- J'adapterais le code de Wapiti afin de générer des modèles capables de stratégies mixtes avec un calcul du taux d'erreurs adapté,
- J'intègrerais les modèles présentés ici dans la partie « modélisation de l'adversaire » d'un agent basé sur l'exploitation.

BIBLIOGRAPHIE

1 Darse Billings, Neil Burch, Aaron Davidson, Robert Holte, Jonathan Schaeffer, Terence Schauenberg, and Duane Szafron, 'Approximating Game-Theoretic Optimal Strategies for Full-Scale Poker', in *IJCAI* (2003), pp. 661-68.

2 É. Borel, and J. Ville, *Applications De La Théorie Des Probabilités Aux Jeux De Hasard* (Jacques Gabay, 1938).

3 Michael Buro, 'The Iwec-2002 Man-Machine Othello Match', (2002).

4 Guillaume Chaslot, Sander Bakkes, Istvan Szita, and Pieter Spronck, 'Monte-Carlo Tree Search: A New Framework for Game Ai', in *AIIDE* (2008).

5 Aaron Davidson, 'Opponent Modeling in Poker: Learning and Acting in a Hostile and Uncertain Environment', (2002).

6 ———, 'Using Artificial Neural Networks to Model Opponents in Texas Hold'em', *Unpublished manuscript* (1999).

7 Andrew Gilpin, Samid Hoda, Javier Peña, and Tuomas Sandholm, 'Gradient-Based Algorithms for Finding Nash Equilibria in Extensive Form Games', in *Internet and Network Economics*, ed. by Xiaotie Deng and FanChung Graham (Springer Berlin Heidelberg, 2007), pp. 57-69.

8 Andrew Gilpin, Tuomas Sandholm, and Troels Bjerre Sørensen, 'Potential-Aware Automated Abstraction of Sequential Games, and Holistic Equilibrium Analysis of Texas Hold'em Poker', in *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE* (Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007), p. 50.

9 Eric Jackson, 'Slumbot: An Implementation of Counterfactual Regret Minimization on Commodity Hardware', (2012).

10 E. T. Jaynes, 'Information Theory and Statistical Mechanics', *Physical Review,* 106 (1957), 620-30.

11 Daphne Koller, Nimrod Megiddo, and Bernhard Von Stengel, 'Fast Algorithms for Finding Randomized Strategies in Game Trees', in *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing* (ACM, 1994), pp. 750-59.

12 Andreas Korol, and Bent Witthold, 'Aipokerbot'2010) <http://pokerbot.workingcopy.de/>.

13 H.W. Kuhn, J. von Neumann, O. Morgenstern, and A. Rubinstein, *Theory of Games and Economic Behavior (Commemorative Edition)* (Princeton University Press, 2007).

14 John Lafferty, Andrew McCallum, and Fernando CN Pereira, 'Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data', (2001).

15 Marc Lanctot, Kevin Waugh, Martin Zinkevich, and Michael Bowling, 'Monte Carlo Sampling for Regret Minimization in Extensive Games', in *Advances in Neural Information Processing Systems* (2009), pp. 1078-86.

16 Thomas Lavergne, Olivier Cappé, and François Yvon, 'Practical Very Large Scale Crfs', in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics* (Uppsala, Sweden: Association for Computational Linguistics, 2010), pp. 504-13.

17 Andrew McCallum, Dayne Freitag, and Fernando CN Pereira, 'Maximum Entropy Markov Models for Information Extraction and Segmentation', in *ICML* (2000), pp. 591-98.
18 P. Morris, *Introduction to Game Theory* (Springer New York, 1994).
19 John F Nash, 'Equilibrium Points in N-Person Games', *Proceedings of the national academy of sciences,* 36 (1950), 48-49.
20 Jonathan Rubin, and Ian Watson, 'Computer Poker: A Review', *Artificial Intelligence,* 175 (2011), 958-87.
21 Jonathan Schaeffer, Robert Lake, Paul Lu, and Martin Bryant, 'Chinook the World Man-Machine Checkers Champion', *AI Magazine,* 17 (1996), 21.
22 Kevin Waugh, Martin Zinkevich, Michael Johanson, Morgan Kan, David Schnizlein, and Michael H Bowling, 'A Practical Use of Imperfect Recall', in *SARA* (2009).
23 Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione, 'Regret Minimization in Games with Incomplete Information', in *Advances in neural information processing systems* (2007), pp. 1729-36.

ANNEXES

Annexe 1 : Fichier log ACPC

```
# name/game/hands/seed 2pl.liacc.unamur_tbr.10.0 hold-
em.limit.2p.reverse_blinds.game 3000 783693058
#--t_response 600000
#--t_hand 600000
#--t_per_hand 7000
STATE:0:cc/cc/crf:6sTs|4hJd/Jc2cKd/9s:-10|10:liacc|unamur_tbr
STATE:1:cc/crf:9d4c|Kh3d/7sAc5c:-10|10:unamur_tbr|liacc
STATE:2:f:2s7h|3d8c:5|-5:liacc|unamur_tbr
…
STATE:2997:crrc/crc/rrc/cc:ThAd|8s8c/TcQcJh/6d/Ks:80|-80:unamur_tbr|liacc
STATE:2998:rc/rc/rc/rrrc:3sQc|Jd4d/Jc6hKs/Qd/4h:-110|110:liacc|unamur_tbr
STATE:2999:crrrc/crf:TdTh|Kd5h/Kc3c9c:-40|40:unamur_tbr|liacc
SCORE:-3255|3255:liacc|unamur_tbr
```

Annexe 2 : Selby Starting Hands Chart

http://www.archduke.org/simplex/art

OPTIMAL HEADS-UP PREFLOP HOLDEM WITH $1-$2 BLINDS WITH $2 BETS

Above the \ diagonal in the table are the suited hands, below unsuited hands. The letters in the table represent a strategy for playing the hand. In all the strategies (except for F) you *never* fold. For the entries marked * you should randomize your play, choosing a strategy (with the given probability) from the choices below the table.

F => Fold
C => Call
R1 => Raise
R2 => Raise, and reraise if raised back.
R3 => Raise, reraise and re-re-raise if raised back.
CR1 => Call-raise
CR2 => Call-raise and reraise if raised back.

SMALL BLIND PLAY

		suited											
		A	K	Q	J	T	9	8	7	6	5	4	3	2
	A	R3	R2	R2	R2	R2	R1	R1	R1	R1	R1	R1	R1	R1
	K	R2	*1	CR1	CR1	CR1	R1	R1	R1	R1	R1	R1	R1	R1
	Q	CR1	CR1	R2	R1	R1	R1	R1	R1	R1	R1	R1	R1	R1
	J	R2	*2	R1	R2	R1	R1	R1	R1	R1	R1	R1	R1	R1
u	T	*3	R1	R1	R1	R2	R1	R1	R1	R1	R1	R1	C	C
n	9	R1	R1	R1	R1	R1	R2	R1	R1	R1	R1	C	C	C
s	8	R1	R1	R1	R1	R1	R1	R2	R1	R1	C	C	C	C
u	7	R1	R1	R1	R1	R1	R1	C	R2	C	C	C	C	C
i	6	R1	R1	R1	R1	R1	C	C	C	R1	C	C	C	C
t	5	R1	R1	R1	*4	C	C	C	C	C	R1	C	C	C
e	4	R1	R1	C	C	C	C	C	C	C	C	R1	C	C
d	3	R1	C	C	C	C	C	F	F	F	C	F	R1	C
	2	R1	C	C	C	C	C	F	F	F	F	F	F	C

[1] (KK) 60.1% R3, 39.9% CR2
[2] (KJo) 62.7% R1, 37.3% CR1
[3] (ATo) 73.0% R2, 27.0% CR1
[4] (J5o) 64.3% C , 35.7% R1

BIG BLIND PLAY

When the small blind has called

		suited											
		A	K	Q	J	T	9	8	7	6	5	4	3	2
	A	R3	R2	R2	R2	R2	R1	R1	R1	R1	R1	R1	R1	R1
	K	R2	*1	R1	R1	R1	R1	R1	R1	R1	R1	R1	R1	R1
	Q	R2	R1	R2	R1	R1	R1	R1	R1	R1	R1	R1	R1	R1
	J	R2	R1	R1	R2	R1	R1	R1	R1	R1	R1	R1	R1	R1
u	T	R1	R1	R1	R1	R2	R1	R1	R1	R1	R1	R1	R1	C
n	9	R1	R1	R1	R1	R1	R2	R1	R1	R1	R1	C	C	C
s	8	R1	R1	R1	R1	R1	R1	R2	R1	R1	C	C	C	C
u	7	R1	R1	R1	R1	R1	R1	*2	*3	C	C	C	C	C
i	6	R1	R1	R1	R1	R1	R1	C	C	R1	C	C	C	C
t	5	R1	R1	R1	R1	R1	C	C	C	C	R1	C	C	C
e	4	R1	R1	R1	R1	C	C	C	C	C	C	R1	C	C
d	3	R1	R1	R1	R1	C	C	C	C	C	C	C	R1	C
	2	R1	R1	R1	C	C	C	C	C	C	C	C	C	R1

[1] (KK) 98.1% R2, 1.9% R3
[2] (87o) 26.0% C, 74.0% R1
[3] (77) 99.8% R1, 0.2% R2

When the small blind has raised

		suited											
		A	K	Q	J	T	9	8	7	6	5	4	3	2
	A	R3	R2	R1	R1	R1	R1	R1	R1	R1	R1	R1	R1	R1
	K	*1	R2	R1	R1	R1	R1	R1	C	C	C	C	C	C
	Q	R1	R1	R2	R1	R1	C	C	C	C	C	C	C	C
	J	R1	R1	R1	R2	C	C	C	C	C	C	C	C	C
u	T	R1	R1	C	C	R1	C	C	C	C	C	C	C	C
n	9	R1	R1	C	C	C	R1	C	C	C	C	C	C	C
s	8	R1	C	C	C	C	C	R1	C	C	C	C	C	C
u	7	R1	C	C	C	C	C	C	R1	C	C	C	C	C
i	6	R1	C	C	C	C	C	C	C	R1	C	C	C	C
t	5	*2	C	C	C	C	C	C	C	C	R1	C	C	C
e	4	C	C	C	C	C	C	C	C	C	C	C	C	C
d	3	C	C	C	C	C	C	C	C	C	C	C	C	C
	2	C	C	C	C	C	C	C	C	C	C	C	C	C

[1] (AKo) 14.2% R1, 85.8% R2
[2] (A5o) 5.4% C, 94.6% R1

Annexe 3 : ACPC Protocol Specification version 2.0.0

This document describes the format of messages between the ACPC dealer program (server) and a player (client.) Communication is over TCP, with clients connecting to the server (the client will be given a numeric address) using the ports printed out by the server. All messages are terminated by a carriage return and a new line ('\r\n' or ASCII characters 13 and 10, respectively.) Any message from the server which starts with '#' or ';' is a comment or GUI command, and may be safely ignored.

The first message from a player must be a version string

```
<version> := 'VERSION:2:0:0\r\n'
```

After this, the server will repeatedly send messages to the client giving their view of the match state, including states where the client is not acting, and the final state when the game is over. If the client is acting, they will respond by returning a message with the same state, followed by an action.

```
<serverMessage> := <matchState> '\r\n'
<matchState> := 'MATCHSTATE:' <position> <handNumber> <betting> <cards>
<position> := <unsigned integer> ':'
<handNumber>:= <unsigned integer> ':'
<betting> := { <limitBetting> } { <nolimitBetting> } ':'
<limitBetting> := ( <round1LimitBetting>
{ | <round1LimitBetting> '/' <round2LimitBetting> … } )
<roundXLimitBetting> = <limitAction>*
<limitAction> := <fold> | <call> | <limitRaise>
<fold> := 'f'
<call> := 'c'
<limitRaise> := 'r'
<nolimitBetting> := ( <round1NolimitBetting>
{ | <round1NolimitBetting> '/' <round2NolimitBetting> … } )
<roundXNolimitBetting> := <noLimitAction>*
<noLimitAction> := <fold> | <call> | <nolimitRaise>
<nolmitRaise> := 'r' <nolimitRaiseSize>
<nolimitRaiseSize> := <unsigned integer>
<cards> := <holeCards> <boardCards>
<holeCards> := <player1Cards> '|' <player2Cards> { '|' <player3Cards>
… }
<boardCards> := <round1BoardCards> { '/' <round2BoardCards> … }
<playerXCards> := '' | <card> { <card> … }
<roundXBoardCards> := { <card> … }
```

```
<card> := <rank> <suit>
<rank> := '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9' | 'T' | 'J' |
'Q' | 'K' | 'A'
<suit> := 's' | 'h' | 'd' | 'c'
<clientResponse> := <matchState> ':' { <limitAction> } { <noLimitAc-
tion> } '\r\n'
```

Parts of the specification in brackets indicate game specific definitions. For example, in Texas Hold'em, there are four rounds with zero, three, one, and one board cards on the rounds respectively, so the actual definition would be

```
<round1BoardCards> := ''
<round2BoardCards> := <card> <card> <card>
<round3BoardCards> := <card>
<round4BoardCards> := <card>
```

The <position> field tells the client their position relative to the dealer button. A value of 0 indicates that for the current hand, the client is the first player after the button (the small blind in ring games, or the big blind in reverse-blind heads-up games.)

The <handNumber> is simply an identifier for the current hand. Clients should make no assumption about these values, other than that it will be unique across hands within a match, and that it will not change within a single hand.

<betting> is a list of actions taken by all players. There is no distinction made between calling and checking, or betting and raising. In no-limit betting strings, the raise action includes a size, which indicates the total number of chips the player will have put into the pot after raising (ie the value they are raising to, not the value they are raising by.) Valid betting strings will be described in a later section.

<cards> is the complete set of cards visible to the player, given their <position>. <holecards> describes the view of the private cards, and the number of <playerCards> sections is determined by the game being player. For example, heads-up games will have two sections, 3 player ring games will have 3 sections, and so on. Each <playerCard> section will either be an empty

string, indicating that the player does not know what those cards are, or a number of <card> strings determined by the game. A <card> is simply two characters, one for the rank and one for the suit. The <boardCards> description will also have a number of sections, one for each round, up to the current round as indicated by the <betting>. The number of <cards> in each section is fixed, and determined by the game. Note that if it ever becomes desirable to play a game with board cards in the first round, the protocol should probably be changed again to add a separator between the hole cards and board cards.

A valid <betting> string consists of a sequence of valid actions for successive acting players. Calling is always valid. Folding is only valid if the acting player would need to add money to the pot to call. To describe raises, it is worth making a distinction between raising by and raising to. Unless this document specifically mention otherwise, when it speaks of a raise size, it is talking about a raise to a value, using the term to mean the total number of chips the player will have put into the pot, including chips added in previous rounds. In contrast, raising by a value is adding that number of chips to the pot after calling the current bet.

A raise is valid if a) it raises by at least one chip, b) the player has sufficient money in their stack to raise to the value, and c) the raise would put the player all-in (they have spent all their chips) or the amount they are raising by is at least as large as the big blind and the raise-by amount for any other raise in the round. This description properly handles some odd cases where some players may be all in. Informally, at the beginning of a round a player may bet as low as the big blind, and each subsequent raise must increase the bet by at least as much as the last raise. In limit betting games, there may also be a limit on the number of raises in a round, and any raise action over this limit is not valid.

An active player is one that has not folded, and is not all-in. Label the players from 0 to N-1, where N is the number of players in the game. After an action by player p, the next acting player is p' for the minimum d>0 such that p' is active and p' =(p+d) modulo N. That is, the next player around the table who has at least two valid actions. The first player in a round is specified by the game.

A betting round ends when all active players remaining have either called, or were the player which initiated the current bet. The game is over if the all betting rounds have finished, or if there are not at least two active players left in the game. If all players but one have folded, the remaining player wins the entire pot, and does not show other players their cards. Otherwise, the game has ended in a showdown and all non-folding players show everyone their cards.

Values in a showdown are determined by the rank of a players best hand, constructed from their hole cards and the public board cards. Folding players have a lower rank than any non-folding player. Every pot of money is split evenly between all participating players with the highest rank. There is a separate pot for every distinct amount j of money spent by a player (whether they fold or not), and a player is participating in the pot if they spent at least that much money (again, whether they fold or not.) The size of the pot is equal to the number of participating players multiplied by (j - the size of the next smallest amount spent by a player).

Changes from Version 1

There a number of changes from the first version of the protocol. Ring games are now documented. No-limit games no longer give a value for a call, and there are no blinds in the string. The question of whether all-in players got to act was previously left unspecified, and different implementations had different behavior. Folding is no longer allowed if the player could have called for free. Finally, this updated description also covers the protocol for no-limit ring games, in the event that this game is added to the competition.

It is also strongly suggested to the chair that on a failed responses from a client, whether the message was malformed, the action was invalid, or the player did not respond in time, the entire match should be ended. If the problem cannot be fixed easily by a very short interaction with the team, the program should be disqualified. The previous choice of having the player call in some cases, or fold all remaining hands led to some situations where the results of the competition depended on the chair's decision on exactly how to handle incorrect behavior. Teams with buggy programs also took up large amounts of the chair's time. As long as players are given sufficient time for testing, using the exact same code used for the competition, it is not unreasonable to

expect players to be able to run without producing error messages or warnings.

Example

The rest of this document is examples of messages to and from a client in various games. Messages from the server are prefaced with S->. Responses from the client are prefaced with <-C.

Two player limit Texas Hold'em

```
S-> MATCHSTATE:0:0::TdAs|
S-> MATCHSTATE:0:0:r:TdAs|
<-C MATCHSTATE:0:0:r:TdAs|:r
S-> MATCHSTATE:0:0:rr:TdAs|
S-> MATCHSTATE:0:0:rrc/:TdAs|/2c8c3h
<-C MATCHSTATE:0:0:rrc/:TdAs|/2c8c3h:r
S-> MATCHSTATE:0:0:rrc/r:TdAs|/2c8c3h
S-> MATCHSTATE:0:0:rrc/rc/:TdAs|/2c8c3h/9c
<-C MATCHSTATE:0:0:rrc/rc/:TdAs|/2c8c3h/9c:c
S-> MATCHSTATE:0:0:rrc/rc/c:TdAs|/2c8c3h/9c
S-> MATCHSTATE:0:0:rrc/rc/cr:TdAs|/2c8c3h/9c
<-C MATCHSTATE:0:0:rrc/rc/cr:TdAs|/2c8c3h/9c:c
S-> MATCHSTATE:0:0:rrc/rc/crc/:TdAs|/2c8c3h/9c/Kh
<-C MATCHSTATE:0:0:rrc/rc/crc/:TdAs|/2c8c3h/9c/Kh:c
S-> MATCHSTATE:0:0:rrc/rc/crc/c:TdAs|/2c8c3h/9c/Kh
S-> MATCHSTATE:0:0:rrc/rc/crc/cr:TdAs|/2c8c3h/9c/Kh
<-C MATCHSTATE:0:0:rrc/rc/crc/cr:TdAs|/2c8c3h/9c/Kh:c
S-> MATCHSTATE:0:0:rrc/rc/crc/crc:TdAs|8hTc/2c8c3h/9c/Kh
S-> MATCHSTATE:1:1::|Qd7c
<-C MATCHSTATE:1:1::|Qd7c:r
S-> MATCHSTATE:1:1:r:|Qd7c
S-> MATCHSTATE:1:1:rr:|Qd7c
<-C MATCHSTATE:1:1:rr:|Qd7c:c
S-> MATCHSTATE:1:1:rrc/:|Qd7c/2h8h5c
S-> MATCHSTATE:1:1:rrc/r:|Qd7c/2h8h5c
<-C MATCHSTATE:1:1:rrc/r:|Qd7c/2h8h5c:c
S-> MATCHSTATE:1:1:rrc/rc/:|Qd7c/2h8h5c/Th
S-> MATCHSTATE:1:1:rrc/rc/r:|Qd7c/2h8h5c/Th
<-C MATCHSTATE:1:1:rrc/rc/r:|Qd7c/2h8h5c/Th:f
S-> MATCHSTATE:1:1:rrc/rc/rf:|Qd7c/2h8h5c/Th
S-> MATCHSTATE:0:2::9d7s|
S-> MATCHSTATE:0:2:r:9d7s|
<-C MATCHSTATE:0:2:r:9d7s|:c
S-> MATCHSTATE:0:2:rc/:9d7s|/5d2cJc
<-C MATCHSTATE:0:2:rc/:9d7s|/5d2cJc:c
S-> MATCHSTATE:0:2:rc/c:9d7s|/5d2cJc
S-> MATCHSTATE:0:2:rc/cc/:9d7s|/5d2cJc/3d
<-C MATCHSTATE:0:2:rc/cc/:9d7s|/5d2cJc/3d:c
```

```
S-> MATCHSTATE:0:2:rc/cc/c:9d7s|/5d2cJc/3d
S-> MATCHSTATE:0:2:rc/cc/cr:9d7s|/5d2cJc/3d
<-C MATCHSTATE:0:2:rc/cc/cr:9d7s|/5d2cJc/3d:f
S-> MATCHSTATE:0:2:rc/cc/crf:9d7s|/5d2cJc/3d
```